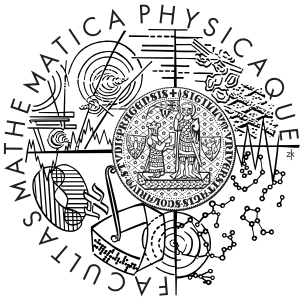
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Pavol Gál

Vlastnosti *k*-intervalových booleovských funkcí
Properties of *k*-interval Boolean functions

Department of Theoretical Computer Science and Mathematical Logic

Supervisor: doc. RNDr. Ondřej Čepek, Ph.D.

Study program: Computer Science, Theoretical Computer Science

2010

I thank my parents and all my friends for their support. I thank Zuzana Vlčková for grammatical and stylistic corrections. Most of all, I am grateful to my supervisor doc. Ondřej Čepek for his dedication, help and advice.

I hereby declare that I wrote the thesis myself using only the referenced sources. I agree with lending and publishing the thesis.

Prague, August 6, 2010 Pavol Gál

Contents

Název práce: Vlastnosti k-intervalových booleovských funkcí
Autor: Pavol Gál
Katedra: Katedra teoretické informatiky a matematické logiky
Vedoucí: doc. RNDr. Ondřej Čepek, Ph.D.
e-mail vedoucího: Ondrej.Cepek@mff.cuni.cz

Abstrakt: Táto práca je zameraná predovšetkým na intervalové booleovské funkcie. Práca prezentuje základné znalosti o booleovských funkciách, ich reprezentáciách a hlavne sa koncentruje na pozitívne booleovské funkcie. Práca cituje viacero známych výsledkov o intervalových funkciách, ako sú ich rôzne vlastnosti, niektoré rozpoznávacie algoritmy a ich zložitosť. Práca ďalej zavádza komutatívne booleovské funkcie a študuje vlastnosti komutatívnych pozitívnych booleovských funkcií a niektorých odvodených foriem. Práca formuluje viacero tvrdení o ich štruktúre a počte intervalov. Novým a najdôležitejším výsledkom je algoritmus na rozpoznávanie pozitívnych 3-intervalových funkcií. Na záver práca analyzuje štruktúru a počet intervalov niektorých konkrétnych všeobecných booleovských funkcií.

Klíčová slova: Booleovské funkcie, intervalové funkcie, komutatívne funkcie, rozpoznávací algoritmus

Title: Properties of k-interval Boolean functions
Author: Pavol Gál
Department: Dep. of Theoretical Computer Science and Mathematical Logic
Supervisor: doc. RNDr. Ondřej Čepek, Ph.D.
Supervisor's e-mail address: Ondrej.Cepek@mff.cuni.cz

Abstract: The main focus of this thesis is on interval Boolean functions. The thesis presents some fundamental knowledge about Boolean functions, their representations and, in particular, concentrates on positive Boolean functions. The thesis quotes several known results about interval functions, such as their various properties, some recognition algorithms and their complexity. Then the thesis introduces commutative Boolean functions and studies the properties of commutative positive Boolean functions and some derived forms. The thesis formulates several propositions about their structure and number of intervals. The most important and new result is the algorithm for recognition of positive 3-interval functions. Finally the thesis analyzes the structure and number of intervals of a few particular general Boolean functions.

Keywords: Boolean functions, interval functions, commutative functions, recognition algorithm

Chapter 1

Introduction

In general, Boolean functions, are very complex topic which has been studied for several decades and introduced many interesting results. Boolean functions can be perceived from several points of view and divided into various classes according to their special characteristics, properties, evaluation complexities etc.

In this thesis we will focus on special class of Boolean functions called Interval functions. We will study their properties and properties of a few specialized subclasses. We will also show some algorithms to recognize functions from these particular subclasses. We will start with some essential definitions. Then we will proceed with the most important representations of Boolean functions, followed by definitions of already mentioned specialized subclasses and then we will finally come to Interval Boolean functions themselves.

1.1 Boolean functions

In this section we will define Boolean functions and the most important terms and notations which are closely related to them.

Definition (Boolean function). A *Boolean function* (or *function* for short) on n propositional variables is a mapping $f\colon \{0,1\}^n \mapsto \{0,1\}$

Boolean values 1 and 0 we usually denote as *true* and *false* respectively.

Definition (Boolean vector). A *Boolean vector* (or *vector* for short) of length n is an n-tuple of such Boolean values. We will denote these vectors by $\boldsymbol{x}$, $\boldsymbol{y}$, ...

For vectors $\boldsymbol{u}$, $\boldsymbol{v}$ of the same length n we denote by $\boldsymbol{u} \geq \boldsymbol{v}$, that $\boldsymbol{u}$ is componentwise greater than or equal to $\boldsymbol{v}$, i.e., $\forall i \in \{1, ..., n\} : u_i \geq v_i$. We use vector relations $=$, $\neq$, $<$, $>$, $\leq$ in a similar manner.

The bits of vector $\boldsymbol{x} \in \{0,1\}^n$ will be denoted by x_1,...x_n. The vector $\boldsymbol{x}$ also corresponds to an integer number x with binary representation equal to $\boldsymbol{x}$. In

this case x_1 is the most significant bit of x and x_n the least significant bit. Hence $x = \sum_{i=1}^{n} x_i 2^{n-i}$. Also, for any integer x we denote by $\boldsymbol{x}$ the vector corresponding to the binary representation of x.

Definition (Truepoint). A Boolean vector $\boldsymbol{x}$ for Boolean function f is called a *truepoint* (or *true* for short) if $f(\boldsymbol{x}) = 1$. The set of all truepoints is denoted by $T(f)$.

Definition (Falsepoint). A Boolean vector $\boldsymbol{x}$ for Boolean function f is called a *falsepoint* (or *false* for short) if $f(\boldsymbol{x}) = 0$. The set of all falsepoints is denoted by $F(f)$.

For function f on n variables and $v \in \{0, 1\}$ we denote by $f[x_i := v]$ the function on $(n-1)$ variables, which is formed from f by fixing the value of i-th variable to v.

Chapter 2

Representations of Boolean Functions

For any progress in the following text we need to represent Boolean functions in a way which is the most suitable for our purposes. We will start with a very simple representation.

2.1 Truth table

Definition (Truth table). A *truth table* for Boolean function f on n variables is the table of all unique n-dimensional input Boolean vectors $\boldsymbol{x}$ of the function f on the left side and their appropriate function values $f(\boldsymbol{x})$ on the right side.

This definition is valid because a Boolean function is defined on a finite set and therefore its truth table has a finite amount of rows. When the function is represented by such a truth table, the input vectors are usually sorted from $\boldsymbol{0}$ vector (vector where each its element is equal to 0) to $\boldsymbol{1}$ vector and variables are separated into columns so each variable has one column. This representation is easy to understand and easy to imagine. In the following text we will use it for easy and visual explanation of other representations. However, since the amount of all input vectors for Boolean function on n variables is 2^n, also its truth table has 2^n rows. Therefore truth table is not an effective representation at all.

2.2 Logical operators

The most common representation of Boolean functions is using *logical operators*. Arguments of such operators can be Boolean variables or results of other logical operators. In other words, logical operators can be (recursively) combined. They differ by number of arguments and are defined by all combinations of Boolean values in arguments and corresponding result values. Such definitions are easily

x_1	x_2	x_3	$f(\boldsymbol{x})$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Table 2.1: Truth table of Boolean function f on 3 variables

figured by truth tables. The most important and the most common operators are negation also denoted by `NOT` or $\neg$, logical conjunction (or conjunction for short) denoted by `AND` or $\wedge$ and logical disjunction (or disjunction) denoted by `OR` or $\vee$. They are all defined in tables 2.2 and 2.3 and we will use them in the rest of the thesis. We added also exclusive disjunction (denoted by `XOR` or $\oplus$). Although this operator is not as commonly used as previous ones, for us it will prove important and we will use it in a few topics.

x	$\neg x$
0	1
1	0

Table 2.2: Logical operators: `NOT`

x	y	$x \wedge y$	$x \vee y$	$x \oplus y$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Table 2.3: Logical operators: `AND`, `OR`, `XOR`

Other logical operators are figured in table 2.4. These are implication ($\Rightarrow$ or $\rightarrow$), equivalence ($\Leftrightarrow$ or $\leftrightarrow$), negation of conjunction (`NAND`) and negation of disjunction (`NOR`). Except of the implication they are all less used operators. The implication is used almost strictly in Propositional and Predicate Logic (see [8]) and `NAND` together with `NOR` are used in circuits (see [13]).

As we mentioned earlier, logical operators are not only unary and binary. For instance, conjunction and disjunction can be defined as n-ary operators as follows.

x	y	$x \Rightarrow y$	$x \Leftrightarrow y$	x NAND y	x NOR y
0	0	1	1	1	1
0	1	1	0	1	0
1	0	0	0	1	0
1	1	1	1	0	0

Table 2.4: Logical operators: implication, equivalence, NAND, NOR

Definition (n-ary conjunction). The *conjunction* of n Boolean variables (or n-dimensional Boolean vector) is true if and only if all n variables (elements of the vector) are true.

Definition (n-ary disjunction). The *disjunction* of n Boolean variables (or n-dimensional Boolean vector) is false if and only if all n variables (elements of the vector) are false.

Observation 2.2.1. Binary conjunction and disjunction as well as exclusive disjunction are commutative and associative binary operators. Which means that for every Boolean variable x, y, z following equalities hold.

$$x \diamond y = y \diamond x \tag{2.1a}$$
$$(x \diamond y) \diamond z = x \diamond (y \diamond z) \tag{2.1b}$$

where symbol $\diamond$ stands for any of previously mentioned operators.

Although these are quite obvious and trivial properties of certain logical operators, we will find them very useful in several chapters of the thesis. At least, these properties allow us to skip brackets when we want to put the same operator (one of these three) between several variables and evaluate them in any order.

Furthermore, this is the reason why iterative use of binary conjunction and disjunction is equivalent with their n-ary definitions.

We have to mention that not all logical operators have properties 2.1. For instance, the implication is neither commutative nor associative operator. Before we continue any further, we will need one more trivial observation.

Observation 2.2.2. Logical conjunction and disjunction are distributive between each other. This means that for every Boolean variable x, y, z equalities 2.2 hold. Furthermore, for negation of these operators De Morgan rules (2.3) hold.

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z) \tag{2.2a}$$
$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) \tag{2.2b}$$

$$\neg(x \wedge y) = \neg x \vee \neg y \tag{2.3a}$$
$$\neg(x \vee y) = \neg x \wedge \neg y \tag{2.3b}$$

2.3 Normal forms and their properties

At this point we can proceed to widely used representations of Boolean functions called **Normal forms**. These representations are based on logical operators `AND`, `OR` and `NOT`. However, before we get to their definition, we need several basic notations and definitions.

For more convenient and compact usage we will denote negation of Boolean variable $\neg x$ as $\overline{x}$. We will also frequently denote conjunction $x \wedge y$ as multiplication xy omitting the operator.

Definition (Literal). A *literal* is Boolean variable x and its negation $\overline{x}$ (*positive* and *negative literal*, respectively). The pair of literals x and $\overline{x}$ is called the *complementary pair*.

Definition (Term). A *term* t is an elementary conjunction of literals if every propositional variable appears in it at most once, i.e., if $I \cap J = \emptyset$

$$t = \bigwedge_{i \in I} x_i \wedge \bigwedge_{j \in J} \overline{x}_j \tag{2.4}$$

A term is called *linear* if it contains exactly one literal and it is called *quadratic* if it contains exactly two literals. We also define analogous construct for disjunction.

Definition (Clause). A *clause* l is an elementary disjunction of literals if every propositional variable appears in it at most once, i.e., if $I \cap J = \emptyset$

$$l = \bigvee_{i \in I} x_i \vee \bigvee_{j \in J} \overline{x}_j \tag{2.5}$$

Definition (Normal forms). A *disjunctive normal form* (or *DNF*) is a disjunction of terms. A *conjunctive normal form* (or *CNF*) is a conjunction of clauses.

Proposition 2.3.1. *Every Boolean function can be represented by a DNF and a CNF.*

Proof. First, we will represent the given function by a truth table. We will now construct the DNF representation. We choose all rows where the function value is 1. From each such row we will construct the term as follows. Each variable

which is set to 1 will appear in the term as positive literal and each variable set to 0 will appear as negative literal. The disjunction of these terms will be the DNF representation of given function. This way we have for each truepoint of the given function exactly one term which will be evaluated as 1. Therefore, for each truepoint disjunction of such terms will be 1. On the other hand, for each falsepoint we have no term which could be evaluated as 1. Hence our DNF will be 0 for each falsepoint of the given function.

The CNF can be constructed from the truth table analogously. For creation of clauses we would just pick the rows with 0 function value and flip the negation of literals. The proof of the equivalence of such CNF with the given Boolean function would be also analogous. ∎

From now on we will from these two representation use only DNF. We will often refer a conjunction of literals (e.g. a part of the term) as set of literals and we will work with it as with set. We can do this because the value of conjunction does not depend on the order of evaluation of its literals but it strictly depends on their presence in it. Also for DNF $\mathcal{F}$ and term t we denote by $t \in \mathcal{F}$ the fact, that t is contained in $\mathcal{F}$.

Definition (Satisfaction, falsification). For Boolean vector v and term t we say, that t *satisfies* (*falsifies* resp.) v if t evaluates to 1 (0 resp.) on v. Analogously we say that $\mathcal{F}$ *satisfies* (*falsifies* resp.) v if $\mathcal{F}$ evaluates to 1 (0 resp.) on v or we can say if at least one term of $\mathcal{F}$ evaluates to 1 (all terms of $\mathcal{F}$ evaluates to 0 resp.) on v.

We define the DNF version of well known *satisfiability problem* (which is originally defined for CNF) as follows.

Definition (Falsifiability problem). Given a DNF $\mathcal{F}$, does there exist an assignment of Boolean values to the variables which is falsified by $\mathcal{F}$?

Definition (Absorption). We say that term t *absorbs* term t' if t consists of a subset of literals which constitute term t'.

Given Boolean functions f and g on the same set of variables, we denote by $f \leq g$ the fact that g is satisfied for any assignment of values to the variables for which f is satisfied. Hence, for example, if a term t absorbs term t', then $t' \leq t$. Also for every term t, which constitutes a term in a DNF $\mathcal{F}$, it holds that $t \leq \mathcal{F}$ since when $t = 1$ for some evaluation of variables, then for the same evaluation $\mathcal{F} = 1$ holds.

Definition (Implicant). We call a term t an *implicant* of a DNF $\mathcal{F}$ if $t \leq \mathcal{F}$. Hence every term $t \in \mathcal{F}$ is an implicant of $\mathcal{F}$.

Definition (Prime implicant). We call t a *prime implicant* if t is an implicant of $\mathcal{F}$ and there is no implicant $t' \neq t$ of $\mathcal{F}$, for which $t \leq t' \leq \mathcal{F}$.

Definition (Prime DNF). We call DNF $\mathcal{F}$ a *prime DNF* if it consists exclusively of prime implicants.

Definition (Irredundant DNF). We call DNF $\mathcal{F}$ *irredundant* if for any term $t \in \mathcal{F}$, DNF $\mathcal{F}'$ produced from $\mathcal{F}$ by deleting t does not represent the same function as $\mathcal{F}$.

Proposition 2.3.2. *If $\mathcal{F}$ belongs to some class of DNFs $\mathcal{C}$, for which we can solve the falsifiability problem in polynomial time and which is closed under partial assignment, then we can test in polynomial time for a term t and a DNF $\mathcal{F}$ whether t is an implicant of $\mathcal{F}$.*

Proof. Let $t = x_1...x_{l_p}\overline{y}_1...\overline{y}_{l_n}$ be a term. Then t is an implicant of $\mathcal{F}$ if and only if $\mathcal{F}[x_1 := 1, ..., x_{l_p} := 1, y_1 := 0, ..., y_{l_n} := 0]$ is not falsifiable (there is no assignment to the remaining variables which makes the DNF evaluate to 0). ∎

Definition (Conflict). We say, that two terms t_1 and t_2 *conflict in a variable x* if t_1 contains literal x and t_2 contains literal $\overline{x}$.

Definition (Consensus). Two terms t_1 and t_2 have a *consensus* if they conflict in exactly one variable. If $t_1 = Ax$ and $t_2 = B\overline{x}$, where A, B are two sets of literals and x is the only variable, in which t_1 and t_2 have conflict, we call a term $t = AB$ a *consensus* of terms t_1 and t_2. We denote this fact by $t = cons(t_1, t_2)$.

Proposition 2.3.3. *If t_1 and t_2 are two implicants of function f, which have a consensus, then $cons(t_1, t_2)$ is an implicant of f.*

Proof. Let $t_1 = Ax$ and $t_2 = B\overline{x}$. If $cons(t_1, t_2) = A \wedge B = 1$, then both $A = 1$ and $B = 1$. Therefore exactly one of the terms Ax and $B\overline{x}$ is equal to 1. And since they are both implicants of f, $f = 1$ must hold. ∎

Given any DNF $\mathcal{F}$, it is possible to create prime and irredundant DNF $\mathcal{F}'$ by certain series of absorptions and consensuses. This method is called *consensual method* and is described in [9]. However, it is not generally very effective and it can easily reach exponential complexity.

The problem of effective creation of prime and irredundant DNF is not the purpose of the thesis. However, there are classes of Boolean functions for which there exists a polynomial solution of this problem.

For more results about logical operators and normal forms we refer to [1] and [10].

2.4 Branching tree

Definition (Branching tree). A *branching tree* of a Boolean function f on n variables is a complete binary tree of depth n. Each inner node represents one variable. The edge which leads to its left (right) son evaluates this variable with 0 (1 respectively). Inner nodes on the same level of the tree represent the same variable and each level corresponds with the different variable. Each path from root to the leaf then represents one of 2^n input Boolean vectors of function f. Each leaf represents the function value, which corresponds to the input vector represented by the path from the root to this leaf.

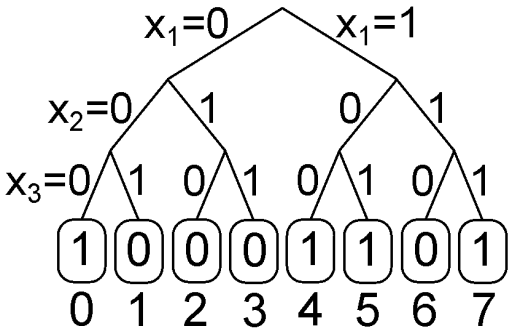


Figure 2.1: A branching tree of the same function as the Truth table 2.1

We can easily observe, that this representation is very large just like the truth table. Actually branching tree and truth table are quite analogous. They both hold all values of the represented function together with their corresponding input vectors. However, the vectors are in the branching tree sorted by its definition, while in the truth table the sorting is optional. Also depending on the order of the variables on the path from the root of the tree to its leaves the information about significance of bits of the vectors is stored in the branching tree. We will find this property extremely important for our purposes. Although, we could easily store such information in truth table as well (representing it by the order of input variable columns and then sorting the input vectors accordingly), we will rather use the branching tree, because it is a graphical representation and therefore it is easier to imagine and remember.

The branching tree has one more property that will be useful for us. Given a Boolean function f and one of its branching tree representations, then by fixing the variable x which is represented by its root to zero (one) we get a sub-function $f[x := 0]$ ($f[x := 1]$, resp.) represented by direct left (right, resp.) subtree of the root.

We define also some operations on branching trees. These operations will transform the branching tree of represented function f and order π of its variables

to another branching tree representing the same function f but different order π' of its variables. We will benefit of these operators later in the thesis.

Definition (Operator moveRootDown(j)). Given the branching tree of function f on n variables and order $\pi = (\pi(1), \pi(2), ..., \pi(n))$ of its variables, operator *moveRootDown(j)* transforms the tree so it represents the function f and order π', where $\pi' = (\pi(2), ..., \pi(j), \pi(1), \pi(j+1), ..., \pi(n))$.

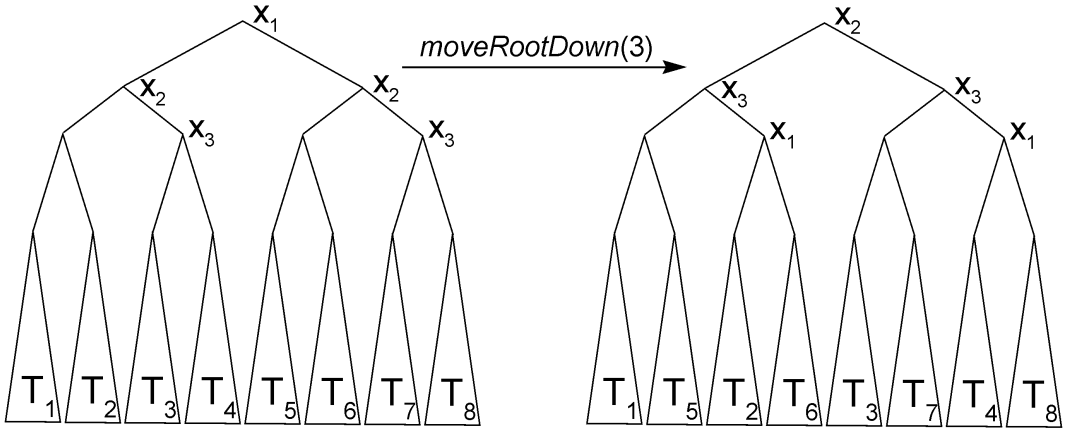


Figure 2.2: Operation *moveRootDown(j)*

This operator simply moves the evaluation of the first (root) variable $j-1$ levels down on the j-th level. Figure 2.2 shows the operator *moveRootDown(j)* and how it transforms the original branching tree. Note, that after applying this operator, all subtrees with roots on j-th level of the remain unchanged, only their order is permuted. In other words, all 2^j functions which can be created from f by fixing first j variables according to the order π are the same, only their representations within the branching tree permute among themselves. Specifically, this permutation of mentioned subtrees can be described as follows. We will take first $\frac{2^j}{2}$ subtrees as they are ordered in the original tree (starting from the leftmost one) and declare them the ordered set L and similarly we declare the ordered set R from the rest of these subtrees finishing in the rightmost one. Now we will create their new ordering by choosing them from these ordered sets, with respect to the set order, starting in the set L and alternating to the other set after each choice.

Definition (Operator moveToRoot(j)). Given the branching tree of function f on n variables and order $\pi = (\pi(1), \pi(2), ..., \pi(n))$ of its variables, operator *moveToRoot(j)* transforms the tree so it represents the function f and order π', where $\pi' = (\pi(j), \pi(1), ..., \pi(j-1), \pi(j+1), ..., \pi(n))$.

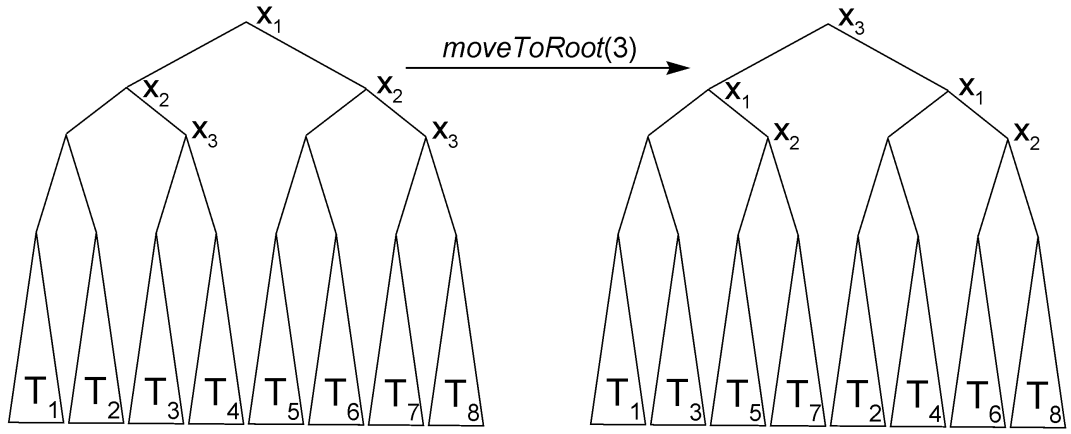


Figure 2.3: Operation *moveToRoot(j)*

This operator moves the evaluation of the variable $x_{\pi(j)}$ $j-1$ levels up to the root of the tree. Figure 2.3 shows, how the operator *moveToRoot(j)* works. We can see that it, again, permutes the subtrees with roots on j-th level. In this case the permutation is as follows. We will take every even subtree in such order as they appear in the original tree and we will will put them it this order after the rest of the subtrees (odd ones), which we will keep in the same relative order as they were in the original tree.

We can easily observe, that previous two operators are reverse each to other. We will now introduce two more operators.

Definition (Operator moveDown(i,j)). Given the branching tree of function f on n variables and order $\pi = (\pi(1), \pi(2), ..., \pi(n))$ of its variables, operator *moveDown(i,j)* transforms the tree so it represents the function f and order π', where $\pi' = (\pi(1), ..., \pi(i-1), \pi(i+1), ..., \pi(i+j), \pi(i), \pi(i+j+1), ..., \pi(n))$.

Definition (Operator moveUp(i,j)). Given the branching tree of function f on n variables and order $\pi = (\pi(1), \pi(2), ..., \pi(n))$ of its variables, operator *moveUp(i,j)* transforms the tree so it represents the function f and order π', where $\pi' = (\pi(1), ..., \pi(i-j-1), \pi(i), \pi(i-j), ..., \pi(i-1), \pi(i+1), ..., \pi(n))$.

Operators *moveDown(i,j)* and *moveUp(i,j)* will take the variable on i-th level of the tree and move it up resp. down j levels. They are, again, reverse each to other. It is also easy to observe that operator *moveDown(i,j)* is actually operator *moveRootDown(j+1)* applied on each subtree with root on i-th level of the original tree. Similarly operator *moveUp(i,j)* is operator *moveToRoot(j+1)* applied on each subtree with root on i-th level. Therefore, in both of these cases, the subtrees with roots on $(i+j)$-th level will remain unchanged but permute accordingly.

Observation 2.4.1. Operator *moveRootDown(2)* is identical with operator *moveToRoot(2)*. They will both swap root variable with the one on second level of the branching tree.

Operator *moveDown(i,1)*is identical with operator *moveUp(i+1,1)*. They will both swap variables on i-th and $(i+1)$-th level of the branching tree.

Chapter 3

Positive and Negative Boolean Functions

Definition (Positivity in variable). We say, that the Boolean function f on n variables is *positive in variable* x_i if for each $(n-1)$-dimensional Boolean vector $\boldsymbol{x'}$ holds: $f[x_i := 0](\boldsymbol{x'}) \leq f[x_i := 1](\boldsymbol{x'})$.

Definition (Negativity in variable). We say, that the Boolean function f on n variables is *negative in variable* x_i if for each $(n-1)$-dimensional Boolean vector $\boldsymbol{x'}$ holds: $f[x_i := 0](\boldsymbol{x'}) \geq f[x_i := 1](\boldsymbol{x'})$.

Definition (Positive and negative Boolean function). We say, that the Boolean function f on n variables is *positive* (*negative*) if it is positive (negative, resp.) in all n variables. The class of positive Boolean functions will be denoted by $\mathcal{C}^+$, the class of negative Boolean functions by $\mathcal{C}^-$.

In most of the papers (e.g. in [4]) we can meet with another definition of positive and negative Boolean function. This definition is closely connected to DNF representation. We will present it as theorem of equivalence with previous definition.

Definition (Positive and negative term). A term t defined by 2.4 is called *positive* if it contains no negative literals (i.e., if $J = \emptyset$) and it is called *negative* if it contains no positive literals (i.e., if $I = \emptyset$).

Definition (Positive and negative DNF). A DNF is called *positive* (*negative*) if it contains only positive (negative, resp.) terms.

Theorem 3.0.2. *A Boolean function is positive (negative) if and only if it has at least one representation by a positive DNF (negative DNF, resp.).*

Proof. For constant Boolean function theorem holds trivially.

(if part) Let f be a Boolean function on $n \geq 1$ variables, which has at least one representation by a positive DNF. Let $\mathcal{F}$ be such representation. Let x be an arbitrary variable. Then x is contained in $\mathcal{F}$ only as positive literal. Let t_1,...,t_k be all terms in $\mathcal{F}$ which contain x and s_1,...,s_l all terms which do not contain x. In $\mathcal{F}[x := 0]$ all terms t_i will disappear and in $\mathcal{F}[x := 1]$ all terms t_i will be shortened by one variable (x). Terms s_j will remain the same in both cases. Therefore all terms which are in $\mathcal{F}[x := 0]$ are also in $\mathcal{F}[x := 1]$, hence for each $\boldsymbol{y} \in \{0,1\}^{n-1}$ $\mathcal{F}[x := 0](\boldsymbol{y}) \leq \mathcal{F}[x := 1](\boldsymbol{y})$ holds.

(only if part) Let f be positive function on $n \geq 1$ variables. Let $\mathcal{F}$ be its DNF. Let x be a variable which is contained in $\mathcal{F}$ as negative literal. Let $t = A\overline{x}$ be an implicant of f, which is present in $\mathcal{F}$. Let us now consider term $t' = Ax$. Let $\boldsymbol{y}$ be a vector which is satisfied by t'. Naturally, $\boldsymbol{y}$ has the bit which corresponds to variable x set to 1. We will create vector $\widetilde{\boldsymbol{y}}$ from $\boldsymbol{y}$ by switching the bit, which corresponds to variable x, from 1 to 0. Then t satisfies $\widetilde{\boldsymbol{y}}$ and since t is implicant f, $\widetilde{\boldsymbol{y}}$ is the truepoint of f.

From positivity of f we know, that $\mathcal{F}(\widetilde{\boldsymbol{y}}) \leq \mathcal{F}(\boldsymbol{y})$, because $\widetilde{\boldsymbol{y}}$ differs from $\boldsymbol{y}$ in exactly one bit and this bit is 1 in $\boldsymbol{y}$. Hence $\boldsymbol{y}$ is also a truepoint and t' is also implicant of f. Therefore if we create DNF $\mathcal{F}'$ from $\mathcal{F}$ by adding term t' (if it was not already present), it represents the same function f. Now we can add implicant $A = cons(t, t') = cons(Ax, A\overline{x})$ which will absorb both terms t and t'.

This way we can simply omit all negative literals in any DNF of positive Boolean function thus creating a positive DNF of this function.

Proof is analogous for negative function. ∎

Proposition 3.0.3. *Class of positive (negative) Boolean functions is closed under partial assignment.*

Proof. Follows trivially from existence of its positive (negative) DNF. By assigning any variable in such DNF the positivity of all remaining literals stays unchanged. ∎

Lemma 3.0.4. *For any positive Boolean function f and its arbitrary variable x holds, that $f = f[x := 0] \vee xf[x := 1]$.*

Proof. We will consider prime and irredundant DNFs $\mathcal{F}$ and $\mathcal{F}$' of function f and the right side of the equation respectively and prove that they are equal. These DNFs are, of course, positive, because as the proof of Theorem 3.0.2 shows, any implicant which contains negative literal is absorbed by implicant without this literal and therefore it cannot be prime.

Let $\mathcal{F}$ consist of prime implicants $T = \{t_1, ..., t_k, xr_1, ..., xr_l\}$, where none of the terms t_i or r_j contains literal x. Any term t_i is not a subset of any term r_j, otherwise r_j would be absorbed. This means that the prime and irredundant DNF $\mathcal{F}_0$ of function $f[x := 0]$ consists exactly of terms $T_0 = \{t_1, ..., t_k\}$, because the rest of the terms are evaluated as false by fixation of variable x to 0.

Similarly, the prime and irredundant DNF $\mathcal{F}_1$ of function $f[x := 1]$ consists of terms $T_1 = \{t_i, r_1, ..., r_l | i \in I \subseteq \{1, ..., k\}\}$ (some terms t_i can be absorbed by some terms r_j, but this does not concern us). Hence $\mathcal{F}'$ consists of implicants $\{t_1, ..., t_k, xr_1, ..., xr_l\} = T$, because any term xt_i is absorbed by corresponding term t_i. ∎

Chapter 4

Interval Boolean Functions

4.1 Basic definitions

Interval Boolean functions were first introduced in [7]. This research was inspired by optimization requirements in the area of hardware verification ([5], [11]) and software testing ([12]), where compact DNF representations of interval functions provide faster generation of the test data.

Before we proceed to the definition of interval functions themselves, we need to define the permutation of a Boolean vector.

Definition (Permutation of vector). For vector $\boldsymbol{x} \in \{0,1\}^n$ and for permutation $\pi : \{1,...,n\} \mapsto \{1,...,n\}$ we denote by $\boldsymbol{x}^\pi$ the vector of n bits formed by permuting bits of $\boldsymbol{x}$ by π. That means $x_i^\pi = x_{\pi(i)}$. By x^π we denote the number with binary representation $\boldsymbol{x}^\pi$.

Definition (0-interval function). Booolean function $f : \{0,1\} \mapsto \{0,1\}^n$ is called a *0-truepoint-interval function* (*0-falsepoint-interval function*) if it is identically equal to zero (one, resp), i.e., $f(\boldsymbol{x}) = 0$ ($f(\boldsymbol{x}) = 1$, resp.) holds for all $x \in \{0,1\}^n$.

Definition (k-truepoint-interval function). Boolean function $f : \{0,1\} \mapsto \{0,1\}^n$ is called a *k-truepoint-interval function* for $k \geq 1$ if it is a $(k-1)$-truepoint-interval function or there exist k pairs of n-bit integers a^1,b^1,...,a^k,b^k, $a^1 \leq b^1 < a^2 \leq b^2 < ... < a^k \leq b^k$, and permutation π of $\{1,...,n\}$, such that for every n-bit vector $\boldsymbol{x} \in \{0,1\}^n$ we get $f(\boldsymbol{x}) = 1$ if and only if $x^\pi \in \cup_{i=1}^k [a^i, b^i]$. The class of k-truepoint-interval functions will be denoted by $\mathcal{C}_{k-T-int}$. The class of positive (negative resp.) k-truepoint-interval functions will be denoted by $\mathcal{C}^+_{k-T-int}$ ($\mathcal{C}^-_{k-T-int}$, resp.).

We define *k-falsepoint-interval function* analogously ($f(\boldsymbol{x}) = 0$ in this case).

In the following text we will usually work with k-truepoint-interval functions. Therefore we will refer to them as k-interval functions omitting "truepoint" to

make it shorter. Also notation of classes of k-truepoint-interval functions will be usually without the "T" character: $\mathcal{C}_{k-int}$, $\mathcal{C}^{+}_{k-int}$ and $\mathcal{C}^{-}_{k-int}$. If we need to distinguish between truepoint and falsepoint k-interval function we will refer to them explicitly.

Now, when we know the definition of interval functions, we can see the usefulness of the branching tree. The order of variables along the path from root to the leaves represents particular permutation π and leaves of the tree, read form left to right, display the studied intervals for this π.

Definition (Commutative Boolean function). We say, that Boolean function f is *commutative* if the vector of leaves of the branching tree is identical for every ordering π of variables of function f.

Example. Boolean function on n variables $\diamond_{i=1}^{n} x_i$ where $\diamond$ stands for one of conjunction, disjunction and exclusive disjunction is commutative. It trivially follows from their properties 2.1. We already know, that $\wedge_{i=1}^{n} x_i$ is actually n-ary conjunction and $\vee_{i=1}^{n} x_i$ is n-ary disjunction. Function $\oplus_{i=1}^{n} x_i$ is called *parity on n variables* and it equals 1 if and only if the sum of all its input variables is odd. We denote it as $\texttt{PARITY}_n$ and we will study this particular function further later in the thesis. We will also study negation of parity on n variables, which we will denote as $\texttt{PARITY}_n \oplus 1$. It is quite clear, that $\texttt{PARITY}_n \oplus 1$ is a commutative function as well and it has the same truepoint intervals as $\texttt{PARITY}_n$ has falsepoint intervals and vice versa.

4.2 Hardness of Recognition problem

Besides studying the properties of k-interval functions, we would also like to look into following problem (or its weaker variations): We are given DNF $\mathcal{F}$ and we want to decide whether $\mathcal{F}$ represents an k-interval function and if so we also want to output the permutation of input bits and the intervals $[a_i, b_i]$ for $i \in \{1, ..., k\}$ which prove that $\mathcal{F}$ represents a k-interval function. This problem is hard when we consider general DNFs (i.e. with no additional requirements) as inputs. This fact is presented more specifically in following theorem, which is a generalized version of Theorem 1 in [2].

Theorem 4.2.1. *It is co-NP-hard to decide whether a given general DNF represents an k-interval function. This remains true even when we are given a permutation π specifying the only permutation of input bits we should consider. In the latter case the problem is co-NP-complete.*

Proof. Let DNF $\mathcal{F}$ be given. Let us first prove the co-NP-hardness. We will take an instance of the problem TAUT which is defined as follows: given a DNF $\mathcal{F}$ decide whether $\mathcal{F}$ is a tautology or, in other words, if $\mathcal{F}$ evaluates to 1 for all

assignments of Boolean values to variables. This problem is known to be co-NP-complete [6]. By deciding whether $\mathcal{F}$ represents a 1-interval function (for some ordering of input bits) and determining corresponding interval $[a, b]$ we can provide answer to the problem TAUT because $\mathcal{F}$ represents tautology if and only if $[a, b]$ equals $[0, 2^n - 1]$ that is $[a, b]$ spans all the integers represented by vectors from $\{0, 1\}^n$. Moreover, for tautology it is not significant which order of variables we consider, tautology is constant 1 on the whole interval $[0, 2^n - 1]$ for every order.

To see that the latter version of our problem is in co-NP we only need to observe that when we are given the permutation π of variables and $2k+1$ binary vectors $\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_{2k}$ such that $\boldsymbol{x}_1 < \boldsymbol{x}_2 < ... < \boldsymbol{x}_{2k}$ and $\mathcal{F}(\boldsymbol{x}_{2i}) = 1$ for $i \in \{0, ..., k\}$ and $\mathcal{F}(\boldsymbol{x}_{2j+1}) = 0$ for $j \in \{0, ..., k-1\}$ we have a certificate of a negative answer which can be verified in polynomial time. ∎

4.3 Closure of k-interval functions

We will now present a few general properties of k-interval functions as they are introduced in [3]. We will use them quite frequently in the rest of the thesis.

Lemma 4.3.1. *Let f be a Boolean function on n variables and let π be an ordering of its variables, with respect to which f can be represented by k intervals. Then the function $f' = f[x_i := c]$, where $i \in \{1, ..., n\}$ and $c \in \{0, 1\}$, can be represented by k or less intervals with respect to ordering π' formed from π by restriction on $\{1, ..., i-1, i+1, ..., n\}$.*

Proof. Let π be the ordering of variables for which f can be represented by k intervals. We will proceed by contradiction. Suppose that f' cannot be represented by k or less intervals with respect to ordering π' which is π projected on $\{1, ..., i-1, i+1, ..., n\}$. Then there are $2k+1$ vectors $\boldsymbol{a}_1, ..., \boldsymbol{a}_{k+1}, \boldsymbol{b}_1, ..., \boldsymbol{b}_k$ of $(n-1)$ bits with the following properties:

1. $\forall j \in \{1, ..., k+1\} : f'(\boldsymbol{a}_j) = 1$
2. $\forall j \in \{1, ..., k\} : f'(\boldsymbol{b}_j) = 0$
3. $\forall j \in \{1, ..., k\} : a_j^{\pi'} < b_j^{\pi'} < a_{j+1}^{\pi'}$

These vectors prove that f' cannot be represented by k intervals with respect to ordering π'. But if we insert the value c as the i-th bit in every vector $\boldsymbol{a}_j$ and $\boldsymbol{b}_l$ we will get $2k+1$ vectors of n bits proving that f cannot be represented by k intervals with respect to the ordering π. This is the desired contradiction. ∎

Corollary 4.3.2. *For any $k \in \{0, ...\}$ the classes $\mathcal{C}_{k-T-int}$, $\mathcal{C}^+_{k-T-int}$, $\mathcal{C}^-_{k-T-int}$ are closed under partial assignment (holds for k-falsepoint-interval functions as well).*

Proof. Follows directly from Lemma 4.3.1 and Proposition 3.0.3. ∎

Chapter 5

Positive k-Interval Functions

In this chapter we will more thoroughly examine positive k-interval functions. We will present some basic properties and already known recognition algorithms. We will also study commutative positive k-interval functions and formulate some theorems about them. And finally we will introduce new algorithm for recognition of positive 3-interval functions. Since construction of positive and prime (and consequently irredundant) DNF of Boolean function is in general hard problem, we will always assume that the given positive function is already in this form.

5.1 Basic properties of positive k-interval functions

Following lemma is already introduced and proved in [3] for positive k-truepoint-interval functions. We will generalize it also for k-falsepoint-interval functions.

Lemma 5.1.1.

(a) Let f be a positive k-truepoint-interval function, where $k \geq 1$, on n variables, which is not a 0-truepoint-interval function and which is represented by intervals $[a_1, b_1] < ... < [a'_k, b'_k]$ where $0 < k' \leq k$. Then $b'_k = 2^n - 1$.

(b) Let f be a positive k-falsepoint-interval function, where $k \geq 1$, on n variables, which is not a 0-falsepoint-interval function and which is represented by intervals $[a_1, b_1] < ... < [a'_k, b'_k]$ where $0 < k' \leq k$. Then $a_1 = 0$.

Proof.

(a) Clearly, if f is positive (and not identically equal to 0, which is implied by the assumption), $f(\mathbf{1}) = 1$ must hold. This follows directly from existence of positive DNF (Theorem 3.0.2).

(b) Analogous. ∎

Analogous version of Lemma 5.1.1 naturally holds for negative functions. We will use this lemma implicitly in the rest of this chapter. But before we proceed, let us present a simple corollary.

Corollary 5.1.2. *For each $k \geq 1$ and non-constant function f holds:*

$$f \in \mathcal{C}^{+}_{k-T-int} \Leftrightarrow f \in \mathcal{C}^{+}_{k-F-int}$$
$$f \in \mathcal{C}^{-}_{k-T-int} \Leftrightarrow f \in \mathcal{C}^{-}_{k-F-int}$$

In other words, positive and negative non-constant functions have equal number of truepoint and falsepoint intervals.

Proof. Follows trivially from the fact, that for $k \geq 1$ the projection of positive function starts as 0 and ends as 1 (for negative starts as 1 and ends as 0), which is exactly what Lemma 5.1.1 claims. ∎

Lemma 5.1.3. *Let f be a positive k-interval function on n variables according to the ordering of variables π represented by intervals $[a_1, b_1] < ... < [a_{k-1}, b_{k-1}] < [a_k, 2^n - 1]$. Let $\mathcal{F}$ be a positive DNF of function f. Then $\mathcal{F}'$ created from $\mathcal{F}$ by switching the positivity of all literals (to negative) is the DNF of negative k-interval function f' represented by intervals $[0, \overline{a}_k] < [\overline{b}_{k-1}, \overline{a}_{k-1}] < ... < [\overline{b}_1, \overline{a}_1]$ with respect to the same ordering of variables π, where $\overline{a}_i$ and $\overline{b}_i$ are formed from a_i and b_i respectively by exchanging zeros to ones and vice versa, i.e. $\overline{a}_i = 2^n - 1 - a_i$ and $\overline{b}_i = 2^n - 1 - b_i$ for $i \in \{1, .., k\}$.*

Proof. Follows directly from the fact that $\mathcal{F}(\boldsymbol{x}) = 1 \Leftrightarrow \mathcal{F}'(\overline{\boldsymbol{x}}) = 1$ if $\overline{\boldsymbol{x}}$ and $\mathcal{F}'$ are formed from $\boldsymbol{x}$ and $\mathcal{F}$ in such a way as is described in lemma. ∎

Theorem 5.1.4. *Let $\mathcal{A}$ be an algorithm which recognizes positive k-interval functions given in their positive DNF in $\Theta(t)$ time. Then algorithm $\mathcal{A}$ can be used to recognize negative k-interval function given in their negative DNF in $\Theta(t)$ time.*

Proof. Follows directly from Lemma 5.1.3. We will use it backwards. Given a negative DNF $\mathcal{F}$ we will switch the literals thus creating a positive DNF $\mathcal{F}'$. Then we will execute the algorithm $\mathcal{A}$ with input $\mathcal{F}'$. If its answer is **NO**, we also output **NO**. If the answer is that $\mathcal{F}'$ represents a positive k-interval function, we also get the permutation and interval and we output the same permutation and the "mirrored" version of the interval according to the Lemma 5.1.3. ∎

Theorem 5.1.4 is already formulated in [2] for positive 1-interval functions. We have just generalized it for positive k-interval functions.

The conclusion of this section is as follows. When studying positive or negative truepoint or falsepoint k-interval functions, it is sufficient to focus only on positive k-truepoint-interval functions. The rest can be easily converted to them, therefore it is not necessary to deal with them any further.

5.2 Recognition of positive 1-interval functions

In this section we will present the algorithm for recognition positive 1-interval functions together with all its details and proofs as they are introduced in [2]. It is crucial for any further progress, that we thoroughly understand all important properties of positive 1-interval functions and how and why the algorithm works. We will start with the most important property of positive 1-interval function.

Lemma 5.2.1. *Let f be a positive interval Boolean function with respect to permutation π of variables which represents interval $[a, 2^n - 1]$, where $a > 0$. Let $c = a - 1 = c^1c^2...c^n$ and let x_i be the most significant variable with respect to π (that is $i = \pi^{-1}(1)$). Let $\mathcal{F}$ be a prime DNF representation of f. Then either x_i is a linear term in $\mathcal{F}$ or x_i appears in every term of $\mathcal{F}$. Moreover $f' = f[x_i := c^1]$ is again an interval function representing interval $[a', 2^{n-1} - 1]$, where $a' - 1 = c^2c^3...c^n$, and $\mathcal{F}[x_i := c^1]$ is a prime DNF representation of f'.*

Proof. As we know from Lemma 5.1.1, any positive 1-interval function represents (possibly empty) interval $[a, 2^{n-1}]$ for some a. In the case $a \leq 2^{n-1}$ for any vector $\boldsymbol{z} \in \{0,1\}^n$ for which the most significant bit of z^π equals 1 it is necessary for $f(\boldsymbol{z})$ to be 1 because $z^\pi \geq 2^{n-1} \geq a$. Hence in $\mathcal{F}$ there must be the positive linear term formed by variable x_i corresponding to this most significant bit otherwise $\mathcal{F}$ is not prime. By fixing x_i to $c^1 = 0$ in $\mathcal{F}$ we just remove x_i from it. The resulting $\mathcal{F}'$ represents function f' and behaves identically as f on vectors with x_i equal to 0. Thus it is an interval function representing $[0c^2...c^n + 1, 2^{n-1} - 1] = [c^2...c^n + 1, 2^{n-1} - 1]$. DNF $\mathcal{F}'$ is again prime and irredundant because we only removed linear term x_i and x_i doesn't occur in any other term of $\mathcal{F}'$.

In the case $a > 2^{n-1}$ function f must be 0 on any vector $\boldsymbol{z} \in \{0,1\}^n$ for which z^π has the most significant bit set to 0 because all these vectors represent numbers smaller than a. Therefore by fixing variable x_i to 0 in $\mathcal{F}$ we must get DNF equal to constant zero. That means that every term of $\mathcal{F}$ must contain the positive literal x_i. By setting x_i to $c^1 = 1$ in $\mathcal{F}$ we get DNF $\mathcal{F}'$ where every term has x_i removed from it. DNF $\mathcal{F}'$ represents function f' which behaves identically as f on vectors with x_i equal to 1. Thus it is again 1-interval function representing interval formed from $[1c^2...c^n + 1, 2^{n-1}]$ by removing the first bit of both boundaries, that is $[c^2...c^n + 1, 2^{n-1} - 1]$. DNF $\mathcal{F}'$ is again prime and irredundant based on the same properties of $\mathcal{F}$. ∎

Observation 5.2.2. In prime and irredundant DNF $\mathcal{F}$ of a positive 1-interval function on $n \geq 2$ variables, there does not exist pair of two variables x_i, x_j $(i \neq j)$ such that x_i occurs in $\mathcal{F}$ as linear term and x_j occurs in every term of $\mathcal{F}$. This is a trivial observation, since the existence of one of these variables directly contradicts with the existence of the other one.

From the previous lemma and its proof we can better understand the structure of positive 1-interval function. This structure also suggest following strategy

how the recognition algorithm should work. In each step we will find the variable which satisfies the conditions of lemma and fix it appropriately thus truncating the constant half of the branching tree. We will now proceed with algorithm itself.

Algorithm 1 - Positive 1-Interval Function Recognition

Input: A nonempty prime and irredundant positive DNF $\mathcal{F}$ on n variables representing function f.

Output: Order $x_1, ..., x_n$ of variables and n-bit number a if $\mathcal{F}$ represents a positive 1-interval function with respect to this order defined by interval $[a, 2^n - 1]$. **NO** otherwise.

1 **if** $\mathcal{F}(\boldsymbol{1}) = 0$ **then** $\mathcal{F}$ is constant 0, output empty interval **endif**
2 **if** $\mathcal{F}(\boldsymbol{0}) = 1$ **then** $\mathcal{F}$ is constant 1, tautology, output $[0, 2^n - 1]$ **endif**
3 $c := 0$ {c will denote the actual value of $a - 1$
4 $b := 0$, $m := 0$
5 **while** $\mathcal{F}$ can be changed by at least one of the following steps
6 **do**
7 **for each** variable y constituting linear term of $\mathcal{F}$
8 **do**
9 $x_{m+1} := y, m := m + 1$
10 $\mathcal{F} := \mathcal{F}[y := 0]$
11 **enddo** {After cycle terminates $\mathcal{F}$ has no linear terms}
12 **for each** variable y that is contained in every term of $\mathcal{F}$
13 **do**
14 $c := c + 2^{n-1-m}, x_{m+1} := y, m := m + 1$
15 $\mathcal{F} := \mathcal{F}[y := 1]$
16 **enddo** {After cycle terminates $\mathcal{F}$ has no variable contained in every term}
17 **enddo** {end of the while cyclus started at line 6}
18 **while** $m < n$ {if $\mathcal{F}$ is an interval function then it does not depend on the
19 remaining variables and hence these variables do not extend the representing
20 interval}
21 **do**
22 $c := c + 2^{n-1-m}$
23 $m := m + 1$
24 **enddo**
25 **if** $\mathcal{F} = \emptyset$
26 **then**
27 $a := c + 1$
28 output $[a, 2^n - 1]$
29 **else**
30 output **NO**
31 **endif**

Theorem 5.2.3. *Previous algorithm correctly recognizes positive prime DNFs representing positive 1-interval functions and for each such function outputs an order of variables and the interval represented by this 1-interval function.*

Proof. The correctness follows from Lemma 5.2.1. When we process one variable y and fix its value we get another DNF which is again prime and irredundant according to Lemma 5.2.1 and represents a positive 1-interval function if and only if the original function was a positive 1-interval function. We also set the most significant bit of the boundary c according to the condition that y fulfilled. Now we can iterate the process because all preconditions of algorithm are satisfied.

If we cannot process all variables of DNF $\mathcal{F}$ before halting we know the condition of Lemma 5.2.1 is not satisfied and thus $\mathcal{F}$ does not represent 1-interval function.

If we can process all variables of $\mathcal{F}$ then it means that a sufficient condition for $\mathcal{F}$ to represent 1-interval function was satisfied but we still need to add 1 bit to c for every variable of f not present in $\mathcal{F}$.

We also note that the order of steps 7 and 12 must be exactly as in our algorithm because in the last iteration of the while loop 5-17 when we always have DNF consisting of just one variable it is necessary to treat it like a linear term and not increase c by one (on line 14) to get the right value of a. ∎

In [2] is also presented exact method how to implement this algorithm in $\Theta(l)$ time. We will present only basic idea of this implementation, since it is not crucial for our progress to know it in details.

After short analysis of algorithm it is clear that algorithm processes at most n iterations and their complexity depends on following operations:

1. Find and remove from $\mathcal{F}$ (some) variable that occurs in every term of $\mathcal{F}$
2. Find and remove from $\mathcal{F}$ (some) linear term y

To implement these operations so that all iterations of the algorithm run $O(l)$ we use three types of data structures: Structure $T(t)$ corresponding to every term t, a structure $V(x)$ for every variable x, and also a structure $L(r)$ for every literal r of DNF $\mathcal{F}$. For every term t we will also remember in $n(t)$ the number of variables in t and for every variable x we will store in $t(x)$ the number of literals formed by x (this equals to the number of terms x is in).

From these unit structures we are able to build in $O(l)$ time a complex structure which consists of various ordered double-linked lists linked also among themselves. In this structure we can find variables which satisfy Lemma 5.2.1 in constant time because the sorting is done according to $n(t)$ (sorting itself can be performed by RadixSort also in $O(l)$ time). Removing the elements from such structure is straightforward and altogether takes the same time as building it. We remind that this is only the idea of implementation and all details together with the proof of linear complexity can be found in [2].

5.3 Commutative positive Boolean functions

Definition (k-subset function on n variables). For $n \geq 1$ and $1 \leq k \leq n$, we call positive function f on n variables *k-subset function* if it has the following prime and irredundant positive DNF representation:

$$\bigvee_{\substack{J \subseteq \{1,\ldots,n\} \\ |J|=k}} \left(\bigwedge_{i \in J} x_i \right) \tag{5.1}$$

In other words all implicants are of size k and they constitute all subsets of variables of this size. We will denote such function as $\mathcal{S}_k^n$.

Observation 5.3.1. $\mathcal{S}_1^n$ is n-ary disjunction and $\mathcal{S}_n^n$ is n-ary conjunction.

Lemma 5.3.2. *Let f be a commutative Boolean function. Let $f(\boldsymbol{x}) = a$, $a \in \{0,1\}$. Then for any permutation π holds that $f(\boldsymbol{x}^\pi) = a$.*

Proof. Let us consider a branching tree of function f and arbitrary order of its variables π_0. In this branching tree the leaf x, which corresponds to the vector $\boldsymbol{x}$ holds the value a.

In the branching tree for function f and ordering π the leaf x^π holds the value a. But from the definition of commutative function, in this branching tree the leaf x holds the value a as well. Therefore $f(\boldsymbol{x}) = f(\boldsymbol{x}^\pi) = a$. ∎

Proposition 5.3.3. *For any $n \geq 1$ and $1 \leq k \leq n$ function $\mathcal{S}_k^n$ is commutative.*

Proof. Let $\boldsymbol{x}$ be a truepoint of function $\mathcal{S}_k^n$, let $\boldsymbol{x}^\pi$ be arbitrary permutation of its bits. There exists at least one term t in $\mathcal{S}_k^n$ such that $t(\boldsymbol{x}) = 1$. According to the definition, this term contains exactly k literals. Therefore there must be at least k bits in $\boldsymbol{x}$ equal to one (these bits must correspond to variables in t but this does not concern us). This means that $\boldsymbol{x}^\pi$ contains also at least k true-bits. We pick any k such bits and those represent certain subset of function variables of size k. But since in $\mathcal{S}_k^n$ are all k-element subsets of variables as terms, there must be also one which satisfies our choice of true-bits. Hence $\mathcal{S}_k^n(\boldsymbol{x}^\pi) = 1$.

Now let $\boldsymbol{x}$ be a falsepoint. $\mathcal{S}_k^n$ contains all k-element subsets of variables as terms. Therefore $\boldsymbol{x}$ must contain less then k true-bits, otherwise it would be satisfied by at least one of the terms. Hence any bitwise permutation $\boldsymbol{x}^\pi$ of $\boldsymbol{x}$ must also contain less then k true-bits and because of this there exists no term in $\mathcal{S}_k^n$ which would satisfy it. ∎

Theorem 5.3.4. *Let f be a commutative positive Boolean function on n variables, which is not a constant function. Then there exists $k \geq 1$ such that f is k-subset function $\mathcal{S}_k^n$.*

Proof. Let $\boldsymbol{x}$ be a truepoint of function f with the least amount of true-bits. We denote this amount k ($k \geq 1$ because f is not constant zero). Since the function is commutative all bitwise permutations of $\boldsymbol{x}$ have to be truepoints (Lemma 5.3.2). Therefore all k-element subsets of function variables have to be present in its positive DNF representation. Let $\mathcal{S}_k^n$ be such representation.

Let $\boldsymbol{y}_1$ be vector with l true-bits and $\boldsymbol{y}_2$ vector with $l+1$ true-bits. If $\boldsymbol{y}_1$ is truepoint, then all its permutations have to be truepoints because of the commutativity. Then there exists truepoint $\boldsymbol{y}_1^\pi$ which differs from $\boldsymbol{y}_2$ in exactly one bit, specifically false-bit in $\boldsymbol{y}_1^\pi$ and true-bit in $\boldsymbol{y}_2$. From the positivity of f, specifically in variable which corresponds to this bit we have that $f(\boldsymbol{y}_1^\pi) \leq f(\boldsymbol{y}_2)$ and therefore $\boldsymbol{y}_2$ has to be truepoint as well.

By applying this induction step from $l = k$ we see that, all vectors which have at least k true-bits are truepoints and we already know that $\mathcal{S}_k^n$ satisfies these vectors.

Since $\boldsymbol{x}$ is the truepoint with the least amount of true-bits any vector with less than k true-bits has to be falsepoint. And when we choose any such vector $\boldsymbol{z}$ we can see that $\mathcal{S}_k^n(\boldsymbol{z}) = 0$ because all terms in $\mathcal{S}_k^n$ are too large to satisfy $\boldsymbol{z}$. Hence $\mathcal{S}_k^n$ is indeed function f. ∎

Theorem 5.3.5. *Boolean function $\mathcal{S}_k^n$, for any $n \geq 1$ and $1 \leq k \leq n$, is represented by $\binom{n-1}{k-1}$ intervals for any ordering of its variables.*

Proof. We will denote number of intervals for function $\mathcal{S}_k^n$ (for any ordering of its variables) as $|\mathcal{S}_k^n|$.

Let us now analyze, what happens if we fix one of the variables in function $\mathcal{S}_k^n$. Without loss of generality we choose x_n. $\mathcal{S}_k^n[x_n := 0]$ will loose all the terms which contain x_n and the rest of them will remain unchanged. Therefore:

$$\mathcal{S}_k^n[x_n := 0] = \bigvee_{\substack{J \subseteq \{1,\ldots,n-1\} \\ |J|=k}} \left(\bigwedge_{i \in J} x_i \right) = \mathcal{S}_k^{n-1}$$

In $\mathcal{S}_k^n[x_n := 1]$ all the terms which contain x_n will be shortened by one variable and become the terms of length $k-1$. These are all $(k-1)$-element subsets of variables $x_1, \ldots, x_{n-1}$. Terms which do not contain x_n will stay unchanged as terms of size k. However, each one of them has at least one subset of size $k-1$ present in the formula as term (because all of them are present) and it will be absorbed by this term. Therefore all terms of size k are redundant and can be removed and we can write $\mathcal{S}_k^n[x_n := 1]$ as follows:

$$\mathcal{S}_k^n[x_n := 1] = \bigvee_{\substack{J \subseteq \{1,\ldots,n-1\} \\ |J|=k-1}} \left(\bigwedge_{i \in J} x_i \right) = \mathcal{S}_{k-1}^{n-1}$$

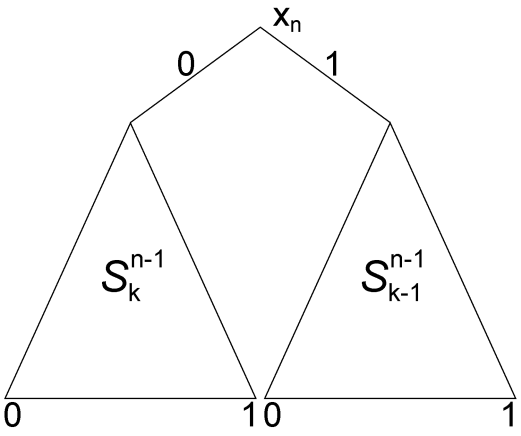


Figure 5.1: A branching tree of function $\mathcal{S}_k^n$

In the branching tree (Figure 5.1) we can see, that the interval projection of $\mathcal{S}_k^n$ is formed by concatenating of projections of $\mathcal{S}_k^n[x_n := 0]$ and $\mathcal{S}_k^n[x_n := 1]$ in this order. And applying Lemma 5.1.1 (i.e. the rightmost leaf in the left subtree is 1 and the leftmost leaf in the right subtree is 0) we can say following:

$$|\mathcal{S}_k^n| = |\mathcal{S}_k^n[x_n := 0]| + |\mathcal{S}_k^n[x_n := 1]| = |\mathcal{S}_k^{n-1}| + |\mathcal{S}_{k-1}^{n-1}|$$

Now for n-ary conjunction and disjunction we know, that they are 1-interval functions. So $|\mathcal{S}_1^n| = 1$ and $|\mathcal{S}_n^n| = 1$ for $n \in \{1, ...\}$. We can also see that for the number of intervals of functions $\mathcal{S}_k^n$ holds the very same recurrent equation as for binomial coefficients $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$. Putting all this knowledge together we get as direct conclusion that $\mathcal{S}_k^n = \binom{n-1}{k-1}$ because binomial coefficients start in $n = k = 0$ and k-subset functions on n variables start in $n = k = 1$. ∎

In [3] a lower bound approximation of number of intervals for $\texttt{MAJORITY}_n$ function is calculated as $2^{\lfloor n/2 \rfloor} - 2$. Boolean function on n variables $\texttt{MAJORITY}_n$ is defined to be 1 if and only if there are more input variables with value 1 than those with value 0. To make this approximation more accurate, we have just calculated the exact number of its intervals.

Proposition 5.3.6. *Boolean function* $\texttt{MAJORITY}_n$ *is represented by* $\binom{n-1}{\lceil n/2 \rceil - 1}$ *intervals for any ordering of its variables.*

Proof. We only need to prove that $\texttt{MAJORITY}_n$ is actually function $\mathcal{S}_{\lceil n/2 \rceil}^n$. Then the proposition follows directly from Theorem 5.3.5.

Let $\boldsymbol{x}$ be a truepoint of $\texttt{MAJORITY}_n$ function. From definition of $\texttt{MAJORITY}_n$ it has to have more true-bits then false-bits. Therefore it has to have at least $\lceil \frac{n}{2} \rceil$ ones. And since $\mathcal{S}_k^n$ has all $\lceil \frac{n}{2} \rceil$-element subsets of its variables as terms, there has to be at least one, which satisfies $\boldsymbol{x}$.

Now let $\boldsymbol{x}$ be a falsepoint. According to the definition of $\texttt{MAJORITY}_n$, it has less than $\lceil \frac{n}{2} \rceil$ ones. In $\mathcal{S}_k^n$ are all terms of length $\lceil \frac{n}{2} \rceil$, hence there is no term which could satisfy $\boldsymbol{x}$. ∎

Corollary 5.3.7. *Maximum number of intervals for a commutative positive Boolean function on n variables is $\binom{n-1}{\lceil n/2 \rceil - 1}$, i.e. this maximum is attained by the $\texttt{MAJORITY}_n$ function.*

Proof. Follows directly form Theorem 5.3.4, Theorem 5.3.5 and the fact, that $\binom{n}{\lfloor n/2 \rfloor} = \binom{n}{\lceil n/2 \rceil}$ is the largest binomial coefficient for given n. ∎

Hypothesis 1. *Let f be a positive k-interval Boolean function on n variables such that $f \in \mathcal{C}^+_{k-int} \setminus \mathcal{C}^+_{(k-1)-int}$. Then $k \leq \binom{n-1}{\lceil n/2 \rceil - 1}$.*

5.4 Forms derived from commutative positive functions

We will start this section with presentation of two lemmas from [3] and their proofs. These lemmas, together with their proofs, will give us better insight into the basic structure of positive k-interval functions.

Lemma 5.4.1. *Let f be a positive k-interval function on n variables. Let i be an index $1 \geq i \geq n$ such that at least one of the following conditions is satisfied.*

$$\forall \boldsymbol{x}(x_i = 0 \Rightarrow f(\boldsymbol{x}) = 0) \tag{5.2a}$$

$$\forall \boldsymbol{x}(x_i = 1 \Rightarrow f(\boldsymbol{x}) = 1) \tag{5.2b}$$

Then there is order π with respect to which f can be represented by l intervals $[c^1, d^1] < ... < [c^l; 2^n - 1]$, where $l \leq k$, and such that $\pi(i) = 1$, i.e., x_i is the most significant variable with respect to π.

Proof. It is best to view the situation on a branching tree. Figure 5.2 captures the branching trees of a function in which index i satisfies the conditions 5.2. We can see that if we take x_i as the most significant variable, then one of the root subtrees contains either only truepoints or only falsepoints. Moreover, since the function considered is positive, the other subtree of the root has the nearest vector also truepoint or falsepoint (same as all the vectors in the other subtree) or it is identically equal to zero or one. Hence the interval representation of this possibly non-constant subtree can be easily extended to an interval representation of the whole tree which has the same number of intervals.

Now we proceed with the formal proof. First, if both conditions 5.2a and 5.2b are satisfied by index i then f does not depend on values of other input variables

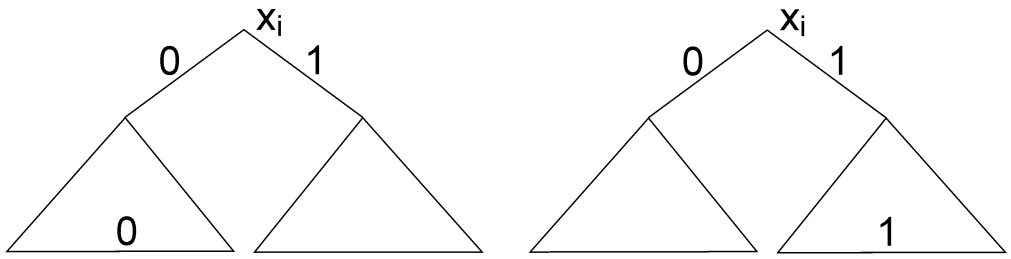


Figure 5.2: Condition 5.2a on the left and condition 5.2b on the right in a branching tree.

but x_i because $f(\boldsymbol{x}) = 1$ for any vector $\boldsymbol{x}$ such that $x_i = 1$ and $f(\boldsymbol{x}) = 0$ for any vector $\boldsymbol{x}$ such that $x_i = 0$. In this case $l = 1$ because f can be represented by interval $[2^{n-1}, 2^n - 1]$ for any order having x_i as the most significant variable.

In the rest of the proof we assume that i satisfies exactly one of conditions 5.2a, 5.2b. We shall define function g on $n-1$ variables. We set $g = f[x_i := 1]$ if i satisfies the condition 5.2a or $g = f[x_i := 0]$ otherwise. In either case g is a k-interval function according to Lemma 4.3.1. Let π' be an order of variables $\{x_1, ..., x_{i-1}, x_{i+1}, ..., x_n\}$ with respect to which g can be represented by $l \leq k$ intervals. We construct the desired order π from π' in the following way. We set $\pi(i) = 1$ and for all $j \neq i$ we set $\pi(i) = \pi'(i) + 1$, i.e., we add x_i as the most significant variable to π'. We claim that f is represented by l intervals with respect to π'. This is true because if the condition 5.2a is satisfied, then all vectors $\boldsymbol{x}$ with $x_i = 0$ are falsepoints, every truepoint $\boldsymbol{u}$ has $u_i = 1$, hence $u^\pi > x^\pi$, and $f(y) = 1$ holds for any vector $\boldsymbol{y}$ with $y_i = 1$ if and only if $g(\boldsymbol{y}') = 1$, where $\boldsymbol{y}$' is formed from $\boldsymbol{y}$ by removing the i-th bit.

If the condition 5.2b is satisfied, then all vectors $\boldsymbol{x}$ with $x_i = 1$ are truepoints, every falsepoint $\boldsymbol{v}$ has $v_i = 0$, hence $v^\pi < x^\pi$, and $f(\boldsymbol{y}) = 0$ holds for any vector $\boldsymbol{y}$ with $y_i = 0$ if and only if $g(\boldsymbol{y}') = 0$, where $\boldsymbol{y}$' is again formed from $\boldsymbol{y}$ by removing the i-th bit. Thus if $[c^1, d^1] < ... < [c^l, 2^{n-1} - 1]$ are the intervals representing g with respect to π' then we can form from them an interval representation of f with respect to π by adding 1 (in case the condition 5.2a is satisfied) or 0 (in case the condition 5.2b is satisfied) as the most significant bit to all interval boundaries except $2^{n-1} - 1$ which is always transformed into $2^n - 1$. ∎

Lemma 5.4.2. *Let $\mathcal{F}$ be a prime DNF representing positive function f. Then*

(a) every term of $\mathcal{F}$ contains variable x_i if and only if i satisfies the condition 5.2a,

(b) variable x_i forms a linear term in $\mathcal{F}$ if and only if i satisfies the condition 5.2b.

Proof. We shall start with the first implications of the equivalences. In (a), if every term of $\mathcal{F}$ contains x_i, then any vector $\boldsymbol{x}$ having $x_i = 0$ is a falsepoint, hence every truepoint must have the i-th bit equal to 1. Similarly, if in (b) variable x_i forms a linear term in $\mathcal{F}$, then any vector $\boldsymbol{x}$ having $x_i = 1$ is a truepoint, hence every falsepoint must have the i-th bit equal to 0.

Now we shall prove the second implications. If i satisfies the condition 5.2a, then by fixing x_i to 0 in $\mathcal{F}$ we must get an empty DNF, because it represents the function identically equal to 0. Hence x_i must occur in every term of $\mathcal{F}$. Similarly, if i satisfies the condition 5.2b, then by fixing x_i to 1 in $\mathcal{F}$, we must get a DNF representing the function, which is identically equal to 1. Hence x_i must form a linear term in $\mathcal{F}$. ∎

Corollary 5.4.3. *Let $\mathcal{F}$ be a positive DNF representing $f \in \mathcal{C}^+_{k-int} \setminus \mathcal{C}^+_{(k-1)-int}$, where $k \geq 1$. Let x_i be a variable forming a linear term in $\mathcal{F}$ (let $c := 0$ in this case), or contained in every term of $\mathcal{F}$ (let $c := 1$ in this case). Then there exists ordering π, with respect to which f can be represented by k intervals and such that $\pi(i) = 1$, i.e., x_i is the most significant variable with respect to π. Moreover, for every ordering π of variables of f, such that $\pi(i) = 1$, f can be represented by k intervals with respect to π if and only if $f[x_i := c]$ can be represented by k intervals with respect to ordering π', where π' is a restriction of π to the remaining variables.*

Proof. Follows directly from Lemmas 5.4.1 and 5.4.2. ∎

Note, that this corollary is generalized Lemma 5.2.1 for k-interval functions and it is presented in [4] as Lemma 2.2.

We will now proceed with some specific, generalized, forms of positive Boolean functions and we will study the properties of these forms. Later, we will be able to use some of them for recognition of positive 3-interval functions.

Definition (k-subset m-variable function form). For $n, m \geq 1$ and $1 \leq k \leq m$, we say that a positive function f on $n + m - 1$ variables $x_1, ..., x_{m-1}, y_1, ..., y_n$ has the *form of k-subset m-variable function* if it can be written as follows:

$$\bigvee_{\substack{I \subseteq \{1,...,m-1\} \\ |I|=k}} \left(\bigwedge_{i \in I} x_i \right) \vee \bigvee_{\substack{J_l \subseteq \{1,...,m-1\} \\ |J_l|=k-1}} \left(\bigwedge_{j \in J_l} x_j \wedge \mathcal{F}_l \right) \tag{5.3}$$

Where $\mathcal{F}_l$ are positive non-constant Boolean functions on n variables $y_1, ..., y_n$. We will denote such form as $\widetilde{\mathcal{S}}^m_k$. Note, that by set $\{1, ..., 0\}$ we mean empty set.

Observation 5.4.4. Number of $\mathcal{F}_l$ functions is $\binom{m-1}{k-1}$, i.e. $l \in \{1, ..., \binom{m-1}{k-1}\}$. This is a direct corollary of the fact that number of $(k-1)$-element subsets of $(m-1)$-element set is $\binom{m-1}{k-1}$.

Observation 5.4.5. $\widetilde{\mathcal{S}}_k^m$ form can be created from $\mathcal{S}_k^m$ function by choice of one arbitrary variable, e.g. x_m, and replacing all its occurrences with n-ary functions $\mathcal{F}_l$. Note, that all terms of $\mathcal{S}_k^m$ which do not contain variable x_m will after such replacement remain unchanged. This also means, that if all functions $\mathcal{F}_l$ are equal and they are functions on one variable, then $\widetilde{\mathcal{S}}_k^m = \mathcal{S}_k^m$.

Observation 5.4.6. $\widetilde{\mathcal{S}}_1^1 = \mathcal{F}_1$. Therefore all positive Boolean functions have form $\widetilde{\mathcal{S}}_1^1$. For $m \geq 2$, functions $\widetilde{\mathcal{S}}_1^m$ and $\widetilde{\mathcal{S}}_m^m$ look like following:

$$\widetilde{\mathcal{S}}_1^m = \bigvee_{\substack{I \subseteq \{1,\dots,m-1\} \\ |I|=1}} \left(\bigwedge_{i \in I} x_i \right) \vee \bigvee_{J_1 = \emptyset} \left(\bigwedge_{j \in J_1} x_j \wedge \mathcal{F}_1 \right) = x_1 \vee x_2 \vee \dots \vee x_{m-1} \vee \mathcal{F}_1$$

$$\widetilde{\mathcal{S}}_m^m = \bigvee_{I = \emptyset} \left(\bigwedge_{i \in I} x_i \right) \vee \bigvee_{J_1 = \{1,\dots,m-1\}} \left(\bigwedge_{j \in J_1} x_j \wedge \mathcal{F}_1 \right) = x_1 \wedge x_2 \wedge \dots \wedge x_{m-1} \wedge \mathcal{F}_1$$

Therefore, if the function has form $\widetilde{\mathcal{S}}_1^m$ or $\widetilde{\mathcal{S}}_m^m$ for certain m, it has also form $\widetilde{\mathcal{S}}_1^{m'}$ or $\widetilde{\mathcal{S}}_{m'}^{m'}$ respectively for any $1 \leq m' \leq m$. Also if the function does not have form $\widetilde{\mathcal{S}}_1^m$ or $\widetilde{\mathcal{S}}_m^m$ for $m \geq 2$, it cannot have form $\widetilde{\mathcal{S}}_1^{m'}$ or $\widetilde{\mathcal{S}}_{m'}^{m'}$ respectively for any $m' \geq m$.

Lemma 5.4.7. *Let f be a positive Boolean function, which can be written in form $\widetilde{\mathcal{S}}_k^m$ for $m \geq 3$ and $2 \leq k \leq m-1$. Then it cannot be written in form $\widetilde{\mathcal{S}}_{k'}^{m'}$ for any $m' \neq m, m' \geq 2$ or any $k' \neq k, 1 \leq k' \leq m'$.*

Proof. Given a form $\widetilde{\mathcal{S}}_k^m$ of function f with some functions $\mathcal{F}_l$, without loss of generality in their prime and irredundant DNF form, let us consider some other form $\widetilde{\mathcal{S}}_{k'}^{m'}$ with some functions $\mathcal{F}_{l'}$ along with prime and irredundant DNF form $\mathcal{D}$ of function f.

We will denote the part of $\widetilde{\mathcal{S}}_k^m$ without functions $\mathcal{F}_l$ as part A and the rest of the formula as part B. For form $\widetilde{\mathcal{S}}_{k'}^{m'}$ it will be A' and B'. Since $m \geq 2$ and $k \leq m-1$, part A is not empty. Also since $m \geq 3$ and $k \geq 2$, all terms in part A are at least quadratic and each function $\mathcal{F}_l$ is conjuncted with at least one literal x_j.

Since all functions $\mathcal{F}_l$ are positive functions on at least one variable, then if we decompose part B using the distributivity, we will get terms of length at least k. Moreover, all terms in part B contain exactly $k-1$ variables from $\{x_1, \dots, x_{m-1}\}$ together with at least one variable y_i, while all terms in part A consist of exactly k literals from set $\{x_1, \dots, x_{m-1}\}$ and no y_i. Therefore no term from part A is a subset of any term from part B and vice versa and consequently no term from part A can absorb any term from part B and vice versa. And since all functions $\mathcal{F}_l$ were in their prime and irredundant DNFs, terms in part B cannot absorb each other as well, therefore this decomposed form of $\widetilde{\mathcal{S}}_k^m$ is already DNF $\mathcal{D}$.

Because all terms in DNF $\mathcal{D}$ are at least of length k, for any form $\widetilde{\mathcal{S}}_{k'}^{m'}$ must hold that $k' \geq k$. If in $\widetilde{\mathcal{S}}_{k'}^{m'}$ held that $k' > k$, then all terms from part A would be too short to be present in part A' of $\widetilde{\mathcal{S}}_{k'}^{m'}$. But since functions $\mathcal{F}_{l'}$ of form $\widetilde{\mathcal{S}}_{k'}^{m'}$ have to contain at least one literal, terms from part A of form $\widetilde{\mathcal{S}}_{k}^{m}$ would be too short also for part B' of form $\widetilde{\mathcal{S}}_{k'}^{m'}$. Therefore $k' \leq k$ and consequently $k' = k$ and $\widetilde{\mathcal{S}}_{k'}^{m'} = \widetilde{\mathcal{S}}_{k}^{m'}$.

Without loss of generality let $V = \{x_1, ..., x_{m'_x}, y_1, ..., y_{m'_y}\}$ be the set of $m' - 1 = m'_x + m'_y$ variables which define form $\widetilde{\mathcal{S}}_{k}^{m'}$.

Of course $m'_x \neq 0$, because in that case we would need to have in part A' terms, which consist exclusively of y_i variables (since $m' > 1$ and consequently $m'_y > 1$) and we do not have such terms in $\mathcal{D}$.

Now let us consider $m'_y = 0$. Naturally, then $m'_x = m' - 1$ and since we only have $m - 1$ variables x_i and for $m' = m$ holds that $\widetilde{\mathcal{S}}_{k}^{m'} = \widetilde{\mathcal{S}}_{k}^{m}$, we only need to analyze the case where $m' < m$, therefore $m'_x < m - 1$.

In part A' we have all necessary terms on variables $x_1, ..., x_{m'_x}$ and the rest of the terms from part A of function $\widetilde{\mathcal{S}}_{k}^{m}$ will be in part B' of $\widetilde{\mathcal{S}}_{k}^{m'}$. Part A' satisfies now the definition of $\widetilde{\mathcal{S}}_{k}^{m'}$. However, in part B' we have, among other terms, term $x_1...x_q x_{m-r}...x_{m-1}T_y$ such that $1 \leq q \leq m'_x$; $m'_x < m - r \leq m - 1$; $q + r = k - 1$ and T_y is a term from one of functions $\mathcal{F}_l$, i.e. there are q literals from $x_1, ..., x_{m'_x}$ and r literals from $x_{m'_x+1}, ..., x_{m-1}$ and some literals y_j. But this term does not satisfy the definition of form $\widetilde{\mathcal{S}}_{k}^{m'}$ because it has less than $k - 1$ literals from set of variables $x_1, ..., x_{m'_x}$. Hence $m'_y \neq 0$.

The last possibility is that $m'_x \geq 1$ and $m'_y \geq 1$. In part A' we have to have all k-element subsets of V as terms. Therefore for $i \in \{0, ..., k\}$ we need each $T_x \in \binom{\{x_1,...,x_{m'_x}\}}{i}$ combined with $T_y \in \binom{\{y_1,...,y_{m'_y}\}}{k-i}$ present in part A' as term $T_x \wedge T_y$. But in DNF $\mathcal{D}$ we have such terms only for $i \in \{k - 1, k\}$. Hence we would need $k = 1$, which is in contradiction with conditions of the lemma. ∎

Lemma 5.4.8. *Let f be a positive Boolean function, which has the form $\widetilde{\mathcal{S}}_{k}^{m}$, where $m \geq 2$ and $2 \leq k \leq m$ with the set of embedded functions $F = \{\mathcal{F}_l; l \in \{1, ..., \binom{m-1}{k-1}\}\}$. Then for any $h \in \{1, ..., m - 1\}$ holds:*

$$\widetilde{\mathcal{S}}_{k}^{m}[x_h := 0] = \widetilde{\mathcal{S}}_{k}^{m-1}$$
$$\widetilde{\mathcal{S}}_{k}^{m}[x_h := 1] = \widetilde{\mathcal{S}}_{k-1}^{m-1}$$

Moreover, the set F_0 of functions $\mathcal{F}_l$ present in $\widetilde{\mathcal{S}}_{k}^{m}[x_h := 0]$ is disjoint with the set F_1 of functions $\mathcal{F}_l$ present in $\widetilde{\mathcal{S}}_{k}^{m}[x_h := 1]$ and $F_0 \cup F_1 = F$.

Proof. Let x_h be arbitrary variable, $h \in \{1, ..., m - 1\}$. We will now analyze, what happens if we fix it.

In function $\widetilde{\mathcal{S}}_k^m[x_h := 0]$ all terms, which contain x_h and all conjunctions with $\mathcal{F}_l$ with occurrence of x_h, will disappear. The rest of the function will remain unchanged. Therefore:

$$\widetilde{\mathcal{S}}_k^m[x_h := 0] = \bigvee_{\substack{I \subseteq \{1,...,h-1,h+1,...,m-1\} \\ |I|=k}} \left(\bigwedge_{i \in I} x_i \right) \vee \bigvee_{\substack{J_l \subseteq \{1,...,h-1,h+1,...,m-1\} \\ |J_l|=k-1}} \left(\bigwedge_{j \in J_l} x_j \wedge \mathcal{F}_l \right)$$

So $\widetilde{\mathcal{S}}_k^m[x_i := 0] = \widetilde{\mathcal{S}}_k^{m-1}$ with $m-2$ variables $x_1, ..., x_{h-1}, x_{h+1}, ..., x_{m-1}$ and certain $\binom{m-2}{k-1}$ functions $\mathcal{F}_l$.

In function $\widetilde{\mathcal{S}}_k^m[x_h := 1]$ all terms, which contain x_h and all $\mathcal{F}_l$ conjunctions with occurrence of x_h, will be shortened by one variable (x_h). All terms without x_h will be absorbed by shortened ones. Also the $\mathcal{F}_l$ conjunctions, without the occurrence of x_h (note, that those are exactly the ones which remained in function $\widetilde{\mathcal{S}}_k^m[x_h := 0]$), will be absorbed by the shortened terms, specifically by the ones without $\mathcal{F}_l$ functions. Therefore $\widetilde{\mathcal{S}}_k^m[x_h := 1]$ will look like following:

$$\widetilde{\mathcal{S}}_k^m[x_h := 1] = \bigvee_{\substack{I \subseteq \{1,...,h-1,h+1,...,m-1\} \\ |I|=k-1}} \left(\bigwedge_{i \in I} x_i \right) \vee \bigvee_{\substack{J_l \subseteq \{1,...,h-1,h+1,...,m-1\} \\ |J_l|=k-2}} \left(\bigwedge_{j \in J_l} x_j \wedge \mathcal{F}_l \right)$$

Hence $\widetilde{\mathcal{S}}_k^m[x_h := 1] = \widetilde{\mathcal{S}}_{k-1}^{m-1}$ with $m-2$ variables $x_1, ..., x_{h-1}, x_{h+1}, ..., x_{m-1}$ and certain $\binom{m-2}{k-2}$ functions $\mathcal{F}_l$, where the set of these functions is disjoint with the set of the ones in $\widetilde{\mathcal{S}}_k^m[x_h := 0]$. ∎

Lemma 5.4.9. *Let f be a positive Boolean function, which has the form $\widetilde{\mathcal{S}}_k^m$, where $m \geq 1$ and $1 \leq k \leq m$ with the set of embedded functions $F = \{\mathcal{F}_l; l \in \{1, ..., \binom{m-1}{k-1}\}\}$. If all functions $\mathcal{F}_l \in F$ are positive 1-interval functions on n variables with respect to the same ordering of their variables $y_{\pi(1)}, ..., y_{\pi(n)}$, then f is a positive $\binom{m-1}{k-1}$-interval function on m+n-1 variables. This holds for any ordering of its variables $x_{\widetilde{\pi}(1)}, ..., x_{\widetilde{\pi}(m-1)}, y_{\pi(1)}, ..., y_{\pi(n)}$, where $\widetilde{\pi}$ is any permutation of $m-1$ elements.*

Proof. We will proceed by induction on n.

We already know that $\widetilde{\mathcal{S}}_1^1 = \mathcal{F}_1$ and for $m \geq 2$ forms $\widetilde{\mathcal{S}}_1^m$ and $\widetilde{\mathcal{S}}_m^m$ look like following:

$$\widetilde{\mathcal{S}}_1^m = x_1 \vee x_2 \vee ... \vee x_{m-1} \vee \mathcal{F}_1$$
$$\widetilde{\mathcal{S}}_m^m = x_1 \wedge x_2 \wedge ... \wedge x_{m-1} \wedge \mathcal{F}_1$$

And if $\mathcal{F}_1$ is a positive non-constant 1-interval function, then both $\widetilde{\mathcal{S}}_1^m$ and $\widetilde{\mathcal{S}}_m^m$ are positive non-constant 1-interval functions according to the iterative use of the Corollary 5.4.3. Moreover, this holds for any ordering of variables $x_1, ..., x_{m-1}$, because all these variables satisfy the conditions of the Corollary 5.4.3 regardless the order of their choice.

Now for the inductive step.

Let $m \geq 2$ and $2 \leq k \leq m-1$. Suppose, that for $l \in \{1, ..., \binom{m-1}{k-1}\}$, we have positive non-constant functions $\mathcal{F}_l$ on n variables $y_1, ..., y_n$, which are 1-interval functions with respect to the same ordering of their variables π. Without loss of generality π is identity. We will, naturally, use the positive DNFs of these functions. We need to prove that if we replace all occurrences of variable x_m in function $\mathcal{S}_k^m$, thus creating the function $\widetilde{\mathcal{S}}_k^m$, it will be positive $\binom{m-1}{k-1}$-interval function for all orderings of its variables which are stated in the lemma.

Let us now consider such function $\widetilde{\mathcal{S}}_k^m$ and analyze it. It is obvious, that this function is positive, because it contains only positive literals and applying the distributivity of binary conjunction and disjunction, which does not change the positivity of literals, we are able to create its positive DNF.

According to the Lemma 5.4.8, $\widetilde{\mathcal{S}}_k^m[x_h := 0] = \widetilde{\mathcal{S}}_k^{m-1}$ and $\widetilde{\mathcal{S}}_k^m[x_h := 1] = \widetilde{\mathcal{S}}_{k-1}^{m-1}$. All functions $\mathcal{F}_l$ are positive 1-interval functions with respect to ordering of their variables $y_1, ..., y_n$. Therefore, from the inductive assumption we have that functions $\widetilde{\mathcal{S}}_k^m[x_h := 0]$ and $\widetilde{\mathcal{S}}_k^m[x_h := 1]$ are positive $\binom{m-2}{k-1}$- and $\binom{m-2}{k-2}$-interval functions respectively with respect to any ordering of variables $x_{\widetilde{\pi}'(1)}, ..., x_{\widetilde{\pi}'(h-1)}$, $x_{\widetilde{\pi}'(h+1)}, ..., x_{\widetilde{\pi}'(m-1)}, y_1, ..., y_n$ ($\widetilde{\pi}'$ being any permutation of m-2 elements). Hence, if we build the branching tree with x_h in root, its left subtree $\widetilde{\mathcal{S}}_k^m[x_h := 0]$ and its right subtree $\widetilde{\mathcal{S}}_k^m[x_h := 1]$ we see, that we have function with $\binom{m-2}{k-1} + \binom{m-2}{k-2} = \binom{m-1}{k-1}$ intervals for any ordering of variables $x_h, x_{\widetilde{\pi}'(1)}, ..., x_{\widetilde{\pi}'(h-1)}$, $x_{\widetilde{\pi}'(h+1)}, ..., x_{\widetilde{\pi}'(m-1)}, y_1, ..., y_n$. And since our choice of x_h was arbitrary, it holds for any ordering $x_{\widetilde{\pi}(1)}, ..., x_{\widetilde{\pi}(m-1)}, y_{\pi(1)}, ..., x_{\pi(n)}$, where $\widetilde{\pi}$ is any permutation of $m-1$ elements. ∎

Now we will allow some of the $\mathcal{F}_l$ functions in form $\widetilde{\mathcal{S}}_k^m$ to be constant. Therefore we need following definition.

Definition ($\widetilde{\mathcal{S}}_k^m[0], \widetilde{\mathcal{S}}_k^m[1]$). Let the positive Boolean function f have the form $\widetilde{\mathcal{S}}_k^m$ and $F = \{\mathcal{F}_l; l \in \{1, .., \binom{m-1}{k-1}\}\}$ be the set of embedded functions. If we allow some functions $F_0 \subseteq F$ to be constant zero (the rest of them will remain non-constant), we will denote the form as $\widetilde{\mathcal{S}}_k^m[0]$. Similarly, if we allow some functions $F_1 \subseteq F$ to be constant one (the rest is again non-constant), we will denote the form as $\widetilde{\mathcal{S}}_k^m[1]$.

Observation 5.4.10. If in previous definition $F_0 = F$, then $\widetilde{\mathcal{S}}_k^m[0] = \mathcal{S}_k^{m-1}$. If $F_1 = F$, then $\widetilde{\mathcal{S}}_k^m[1] = \mathcal{S}_{k-1}^{m-1}$.

Observation 5.4.11. Lemma 5.4.8 holds for forms $\widetilde{\mathcal{S}}_k^m[0]$ and $\widetilde{\mathcal{S}}_k^m[1]$ as well, i.e. for any x_h and $a \in \{0,1\}$ holds:

$$\widetilde{\mathcal{S}}_k^m[a][x_h := 0] = \widetilde{\mathcal{S}}_k^{m-1}[a]$$
$$\widetilde{\mathcal{S}}_k^m[a][x_h := 1] = \widetilde{\mathcal{S}}_{k-1}^{m-1}[a]$$

Lemma 5.4.12.

(a) Let $m \geq 2$ and $1 \leq k \leq m-1$. Let the positive Boolean function f have form $\widetilde{\mathcal{S}}_k^m[0]$, for a set of constant zero functions F_0. Then f is non-constant function, which cannot be represented by less than $\binom{m-2}{k-1}$ intervals for any ordering of its variables.

(b) Let $m \geq 2$ and $2 \leq k \leq m$. Let the positive Boolean function f have form $\widetilde{\mathcal{S}}_k^m[1]$, for a set of constant one functions F_1. Then f is non-constant function, which cannot be represented by less than $\binom{m-2}{k-2}$ intervals for any ordering of its variables.

Proof. We will proceed by induction on n.

(a) For $m \geq 2$ and $1 \leq k \leq m-1$ all functions $\widetilde{\mathcal{S}}_k^m$ contain at least one term, which is not conjuncted with any function $\mathcal{F}_l$, therefore they cannot be constant even if some functions $\mathcal{F}_l$ were constant zero. Hence for $m \geq 2$ and $1 \leq k \leq m-1$ function $\widetilde{\mathcal{S}}_k^m[0]$ is non-constant, thus at least 1-interval for any ordering of its variables. Specifically, this holds for functions $\widetilde{\mathcal{S}}_1^m[0]$ and $\widetilde{\mathcal{S}}_{m-1}^m[0]$, where $m \geq 2$.

Given the function f in form $\widetilde{\mathcal{S}}_k^m[0]$, where $m \geq 3$ and $2 \leq k \leq m-2$, and ordering π of its variables, we want to prove that it has at least $\binom{m-2}{k-1}$ intervals with respect to the ordering π.

Let variables $y_i, ..., y_{i+j}, x_k$ be the most significant variables in this ordering with x_k being the first occurrence of variable from $x_1, ..., x_{m-1}$. Note that j can be equal to zero (thus x_k being the only variable in this list). We will fix all variables $y_i, ..., y_{i+1}$ to 0. After these fixations, the function f is still in the form $\widetilde{\mathcal{S}}_k^m[0]$. Only thing which could happen is, that some more functions $\mathcal{F}_l$ are now constant 0 as well.

Now we are looking at the leftmost subtree in depth j of the branching tree of the given function f with the decision variable x_h. Function still has the form $\widetilde{\mathcal{S}}_k^m[0]$, therefore the direct left subtree of x_h represents function $\widetilde{\mathcal{S}}_k^m[0][x_h := 0] = \widetilde{\mathcal{S}}_k^{m-1}[0]$ and the direct right subtree of x_h function $\widetilde{\mathcal{S}}_k^m[0][x_h := 1] = \widetilde{\mathcal{S}}_{k-1}^{m-1}[0]$. And from the inductive assumption we know, that these functions cannont be represented by less than $\binom{m-3}{k-1}$ and $\binom{m-3}{k-2}$ intervals respectively. Therefore the branching tree in the root x_h cannot represent less than $\binom{m-3}{k-1} + \binom{m-3}{k-2} = \binom{m-2}{k-1}$ intervals with respect to any ordering of remaining variables. Hence the whole branching tree for ordering π cannot represent less than $\binom{m-2}{k-1}$ intervals.

(b) For $m \geq 2$ and $2 \leq k \leq m$ all functions $\widetilde{\mathcal{S}}_k^m$ contain at least one non-empty set of literals, which is conjuncted with corresponding function $\mathcal{F}_l$, therefore they cannot be constant even if some functions $\mathcal{F}_l$ were constant one. Hence for $m \geq 2$ and $2 \leq k \leq m$ function $\widetilde{\mathcal{S}}_k^m[1]$ is non-constant, thus at least 1-interval for any ordering of its variables. Specifically, this holds for functions $\widetilde{\mathcal{S}}_2^m[1]$ and $\widetilde{\mathcal{S}}_m^m[0]$, where $m \geq 2$.

The inductive step for part (b) is analogous with the one in part (a), we only fix variables $y_i, ..., y_{i+j}$ to 1. ∎

Lemma 5.4.13. *Positive Boolean function f, which has form $\widetilde{\mathcal{S}}_k^m$ ($m \geq 1, 1 \leq k \leq m$), is non-constant and cannot be represented by less than $\binom{m-1}{k-1}$ intervals for any set of functions $\mathcal{F}_l$ and any ordering of its variables.*

Proof. We will again proceed by induction on n.

Functions $\mathcal{F}_l$ are non-constant functions, therefore function $\widetilde{\mathcal{S}}_k^m$ is not constant thus at least 1-interval function with respect to any ordering of its variables for any $m \geq 1$ and $1 \leq k \leq m$. Specifically, functions $\widetilde{\mathcal{S}}_1^m$ and $\widetilde{\mathcal{S}}_m^m$ for $m \geq 1$ are at least 1-interval functions for any ordering of their variables.

Let the function f in form $\widetilde{\mathcal{S}}_k^m$, where $m \geq 2$ and $2 \leq k \leq m-1$, be given. We consider arbitrary ordering π of its variables and explore the branching tree. Let variables $y_i, ..., y_{i+j}, x_h$ be the most significant variables in this order according to π with x_k being the first occurrence of variable from $x_1, ..., x_{m-1}$. Note that j can be equal to zero (thus x_h being the only variable in this list).

(L) We will fix these variables to 0 one by one, starting with y_i. Let y_q be the current variable. If there is l such that $\mathcal{F}_l = y_q$, then by fixing it to 0, the analyzed function will gain the form $\widetilde{\mathcal{S}}_k^m[0]$. And according to the Lemma 5.4.12, it cannot be represented by less then $\binom{m-2}{k-1}$ intervals with respect to any ordering of its remaining variables. If such y_q does not occur all the way until we get to x_h, it only means, that we still have the form $\widetilde{\mathcal{S}}_k^m$. According to the Lemma 5.4.8, if we fix x_h to 0, we will get a function in form $\widetilde{\mathcal{S}}_k^{m-1}$. And from the inductive assumption we have that it cannot be represented by less then $\binom{m-2}{k-1}$ intervals with respect to any ordering of remaining variables.

This means that in the branching tree, which represents the given function f and ordering of variables π, exists the leftmost subtree somewhere in the depth $d \in 1, ..., j+1$ which has at least $\binom{m-2}{k-1}$ intervals.

(R) Now we will start to fix variables $y_i, ..., y_{i+j}, x_h$ to 1. Again, let y_q be the current variable. If there is l such that $\mathcal{F}_l = y_q$, then by fixing it to 1, the analyzed function will gain the form $\widetilde{\mathcal{S}}_k^m[1]$. And according to the Lemma 5.4.12, it cannot be represented by less then $\binom{m-2}{k-2}$. If such y_q does not occur until we get to x_h, we still have the form $\widetilde{\mathcal{S}}_k^m$. According to the Lemma 5.4.8, if we fix x_h to 1, we will get a function in form $\widetilde{\mathcal{S}}_{k-1}^{m-1}$. And from the inductive assumption we have that it cannot be represented by less then $\binom{m-2}{k-2}$ intervals with respect to any ordering of remaining variables.

This means that in the analyzed branching tree exists the rightmost subtree in the depth $d \in 1, ..., j+1$ which has at least $\binom{m-2}{k-2}$ intervals.

And when we put (L) and (R) together, we get the conclusion, that the branching tree for function f and ordering of its variables π represents at least $\binom{m-2}{k-1} + \binom{m-2}{k-2} = \binom{m-1}{k-1}$ intervals. ∎

Proposition 5.4.14. *For $m \geq 3$ and $a \in \{0,1\}$ forms $\widetilde{\mathcal{S}}_2^m[a]$ and $\widetilde{\mathcal{S}}_{m-1}^m[a]$ degenerate into forms $\widetilde{\mathcal{S}}_2^{m'}$ and $\widetilde{\mathcal{S}}_{m'-1}^{m'}$ respectively, where $m' < m$.*

Proof. We will denote the part of the formula without functions $\mathcal{F}_l$ as part A and the rest as part B. Let x_j be arbitrary variable from $x_1, ..., x_{m-1}$.

(a) Let the function f have the form $\widetilde{\mathcal{S}}_2^m$. In part A of this form there are $\binom{m-1}{2}$ quadratic terms, specifically $\binom{m-2}{2}$ terms without x_j and $\binom{m-1}{1} = m-1$ terms with x_j. Part B consists of conjunctions $x_l \wedge \mathcal{F}_l, l \in 1, ..., m-1$.

If one of the functions $\mathcal{F}_l$, let it be $\mathcal{F}_j$, is constant 0, the function f will loose conjunction $x_j \wedge \mathcal{F}_j$ and we can rearrange function f as follows. We will move all terms with x_j from the part A to the part B and using distributivity we will create new conjunctions $x_l \wedge (x_j \vee \mathcal{F}_l), l \in 1, ..., j-1, j+1, ..., m-1$ thus creating the form $\widetilde{\mathcal{S}}_2^{m-1}$.

If function $\mathcal{F}_j$ is constant 1, the function f will loose conjunction $x_j \wedge \mathcal{F}_j$ leaving only x_j as linear term. This term will absorb all terms containing x_j of the part A and we will get the form $x_j \vee \widetilde{\mathcal{S}}_2^{m-1}$. So in this case, the function f does not degenerate directly but it creates disjunction of degenerated form and linear term. However, this does not concern us, because in potential recognition algorithm we can at this point use the Corollary 5.4.3.

(b) Now let the function f have the form $\widetilde{\mathcal{S}}_{m-1}^m$. In the part A of this form there is only term $x_1x_2...x_{m-1}$. The part B consists of conjunctions $\bigwedge_{x_i \neq x_l} x_i \wedge \mathcal{F}_l, l \in 1, ..., m-1$.

If function $\mathcal{F}_j$ is constant 0, the function f will loose conjunction $\bigwedge_{x_i \neq x_j} x_i \wedge \mathcal{F}_j$ and x_j is now in all members of the part B as well as in the only term of the part A. Therefore, after using the distributivity, f has the form $x_j \wedge \widetilde{\mathcal{S}}_{m-2}^{m-1}$. So in this case, the function f does not degenerate directly as well but it creates conjunction of degenerated form and literal x_j.

If function $\mathcal{F}_j$ is constant 1, the function f will loose conjunction $\bigwedge_{x_i \neq x_l} x_i \wedge \mathcal{F}_l$ leaving there term only $\bigwedge_{x_i \neq x_l}$. This term will move to the part A and absorb the original term $x_1x_2...x_{m-1}$. All members of the part B can now be rearranged as $x_l \wedge (x_j \wedge \mathcal{F}_l), l \in 1, ..., j-1, j+1, ..., m-1$. Hence the function will gain form form $\widetilde{\mathcal{S}}_{m-2}^{m-1}$ and in potential recognition algorithm we can again use the Corollary 5.4.3.

For all four cases holds that, if any other function $\mathcal{F}_l$ is constant, function f will further degenerate in the corresponding way. ∎

Observation 5.4.15. Such degeneration does not necessarily happen in forms $\widetilde{\mathcal{S}}_k^m$ for $m \geq 5$ and $3 \leq k \leq m-2$.

For instance, if we take the function $\mathcal{S}_3^5$ on variables $x_1, ..., x_5$, we can see that it has the form $\widetilde{\mathcal{S}}_3^5$, where $\mathcal{F}_1 = ... = \mathcal{F}_6 = x_5$. When we now create the form $\widetilde{\mathcal{S}}_3^5[0]$ by replacing the function $\mathcal{F}_6$ with constant zero, we can see that we lost exactly one term and nothing can be absorbed. This function does not have a form $\widetilde{\mathcal{S}}_k^m$ for any $k > 1$ or $m > 1$.

Similarly, when we replace the function $\mathcal{F}_6$ with constant one, thus creating the form $\widetilde{\mathcal{S}}_3^5[1]$, we will get one quadratic term, which will absorb exactly two of the other terms. Such function again does not have a form $\widetilde{\mathcal{S}}_k^m$ for any $k > 1$ or $m > 1$.

In following text we will denote the number of intervals of function f with respect to the ordering of its variables π as $|f|_\pi$.

Lemma 5.4.16. *Let f be a Boolean function on $m + n - 1$ variables, which has form $\widetilde{\mathcal{S}}_k^m$ for $m \geq 1$ and $1 \leq k \leq m$ with the set of embedded functions $F = \{\mathcal{F}_l; l \in \{1, ..., \binom{m-1}{k-1}\}\}$ and let $\widetilde{\pi}$ be ordering of variables $y_1, ..., y_n$. Then for any ordering π of all variables of function f, such that all variables x_i ($i \in \{1, ..., m-1\}$) are more significant then all variables y_j ($j \in \{1, ..., n\}$) and $\widetilde{\pi}$ is the restriction of π on variables $y_1, ..., y_n$, function f has K intervals with respect to the ordering π, where K is following:*

$$K = \sum_{l=1}^{\binom{m-1}{k-1}} |\mathcal{F}_l|_{\widetilde{\pi}} \tag{5.4}$$

Proof. This is a variant of the Lemma 5.4.9 and the proof is very similar. We will again proceed by induction on n.

For $\widetilde{\mathcal{S}}_1^1$ holds trivially. For $\widetilde{\mathcal{S}}_1^m$ and $\widetilde{\mathcal{S}}_m^m$, where $m \geq 2$ holds because of the very same reasons as in proof of the Lemma 5.4.9.

Now let $m \geq 2$, $2 \leq k \leq m-1$ and $h \in \{1, ..., m-1\}$. $\widetilde{\mathcal{S}}_k^m[x_{\pi(h)} := 0] = \widetilde{\mathcal{S}}_k^{m-1}$ and $\widetilde{\mathcal{S}}_k^m[x_{\pi(h)} := 1] = \widetilde{\mathcal{S}}_{k-1}^{m-1}$ with the disjoint sets F_0 and F_1 of functions $\mathcal{F}_l$ such that $|F_0| = \binom{m-2}{k-1}$, $|F_1| = \binom{m-2}{k-2}$ and $F_0 \cup F_1 = F$.

From the inductive assumption we have that $\widetilde{\mathcal{S}}_k^{m-1}$ is represented by $K_0 = \sum_{\mathcal{F}_l \in F_0} |\mathcal{F}_l|_{\widetilde{\pi}}$ and $\widetilde{\mathcal{S}}_{k-1}^{m-1}$ by $K_1 = \sum_{\mathcal{F}_l \in F_1} |\mathcal{F}_l|_{\widetilde{\pi}}$ intervals with respect to any ordering π' on the rest of the variables, which satisfies the conditions of the lemma.

Therefore, the number of intervals of the function f with respect to the ordering π is $K_0 + K_1$ which is exactly what formula 5.4 says. ∎

Hypothesis 2. *Let f be a Boolean function, which can be written in form $\widetilde{\mathcal{S}}_k^m$ for $m \geq 1$ and $1 \leq k \leq m$. Then $f \in \mathcal{C}_{K-int}^+ \setminus \mathcal{C}_{(K-1)-int}^+$ for following K:*

$$K = \min_\pi \left\{ \sum_{l=1}^{\binom{m-1}{k-1}} |\mathcal{F}_l|_\pi \right\}$$

Observation 5.4.17. Given a positive Boolean function in form $\widetilde{\mathcal{S}}_k^m$ on variables $x_1, ..., x_{m-1}, y_1, ..., y_n$ if we fix all variables $x_1, ..., x_{m-1}$ we will get following. If we fix any $k_1 < k-1$ of them to 1 and the rest to 0, then we will get the constant zero function, because there is no term which has less then k variables from this set. If we fix any $k_2 \geq k$ of them to 1 and the rest to zero we will get constant one function, because at least one term from the part A (again, part A are terms without $\mathcal{F}_l$ functions) is evaluated as true. If we fix exactly $k-1$ of them to 1 and the rest to 0 (there is $\binom{m-1}{k-1}$ such fixations) we will get all the functions $\mathcal{F}_l$. This holds because for each such fixation all terms from part A are evaluated as false and exactly one conjunction with one of the $\mathcal{F}_l$ functions is evaluated as true.

Lemma 5.4.18. *Let f be the positive Boolean function on $n+m-1$ variables. If f has the form $\widetilde{\mathcal{S}}_k^m$ ($m \geq 3; 2 \leq k \leq m-1$), then all the sets of $m-1$ variables which define this form have $m-2$ variables shared.*

Proof. Let us consider two arbitrary not equal sets of such $m-1$ variables $X = \{x_1, ..., x_{m-1}\}$ and $X' = \{x'_1, ..., x'_{m-1}\}$. In order to have the form $\widetilde{\mathcal{S}}_k^m$ for both these sets, function f has to contain all k-element subsets of X (part A of the DNF) and all k-element subsets of X' (part A') as terms. But it must also hold, that each term of f contains at least $k-1$ variables from the set X, specifically also the terms in part A'. Since X and X' are not equal, there has to be at least one term in part A' which has at least one variable from set $\widetilde{X}$ (and not from set X).

Without loss of generality, let the term $t_1 = x'_1 x_2 ... x_k$ be one of such terms. Variables $x_2, ..., x_k$ are from set X' and also from set X and x'_1 is element of X' but not of X. Now let us consider any other such term $t_2 = x'_i x_j ... x_{j+k-2}$ (x'_i being the variable from $X' \setminus X$). If it held that $x'_i \neq x_1$, then there had to be term $t_3 = x'_1 x'_i x_j ... x_{j+k-3}$ (if $k = 2$, then $\{x_j, ..., x_{j+k-3}\}$ is empty set) which would have less then $k-1$ variables from X.

Hence $x'_i = x'_1$ and consequently $X' \setminus X = \{x'_1\}$, i.e. sets X and X' have $m-2$ elements shared. Furthermore, all terms contain $k-1$ variables from $X \cap X'$ and for any other set of variables X'', defining the form $\widetilde{\mathcal{S}}_k^m$, must hold that $(X \cap X') \subset X''$. The proof of this inclusion is analogous to the one we have just presented. ∎

Lemma 5.4.18 tells us that if the positive Boolean function f has form $\widetilde{\mathcal{S}}_k^m$, there can be several sets of $m-1$ variables which define this form. However, they all share the same $m-2$ variables. Direct corollary of this fact is that, given the function f which has the form $\widetilde{\mathcal{S}}_k^m$ (for $m \geq 2, 2 \leq k \leq m-1$), it can be also written in following form.

$$\bigvee_{\substack{I\subseteq\{1,\ldots,m-2\}\\|I|=k}}\left(\bigwedge_{i\in I}x_i\right)\vee\bigvee_{\substack{I\subseteq\{1,\ldots,m-2\}\\|I|=k-1\\z\in\{z_1,\ldots,z_r\}}}\left(\bigwedge_{i\in I}x_i\wedge z\right)\vee$$

$$\vee\bigvee_{\substack{J_l\subseteq\{1,\ldots,m-2\}\\|J_l|=k-1}}\left(\bigwedge_{j\in J_l}x_j\wedge\mathcal{G}_l\right)\vee\bigvee_{\substack{J_l\subseteq\{1,\ldots,m-2\}\\|J_l|=k-2}}\left(\bigwedge_{j\in J_l}x_j\wedge\bigwedge_{q=1}^{r}z_q\wedge\mathcal{G}_l\right) \tag{5.5}$$

Where sets $X_i=\{x_1,...,x_{m-2},z_i\}, i\in\{1,...,r\}$ are all unequal sets of variables, which define form $\widetilde{\mathcal{S}}_k^m$ and $\mathcal{G}_l$ are positive functions not containing any of the variables $x_1,...,x_{m_2},z_1,...,z_r$. Note, that for $m=3$ and $k=2$, this is the form (3) presented in [4].

The structure of the form 5.5 is exactly as above because for every set X_i following conditions hold:

1. there have to be all k-element subsets of X_i present as terms
2. in every term there is $(k-1)$-element subset of X_i, specifically either $k-1$ elements from set $\{x_1,...,x_{m-2}\}$ or $k-2$ elements from set $\{x_1,...,x_{m-2}\}$ and variable z_i
3. function has neither the form $\widetilde{\mathcal{S}}_1^2$ nor $\widetilde{\mathcal{S}}_2^2$

We will finish this section with the lemma saying something important about number of intervals of functions which have the form $\widetilde{\mathcal{S}}_k^m$ for more then one set of variables $x_1,...,x_{m-1}$.

Lemma 5.4.19. *Let f be a positive Boolean function on $n+m-1$ variables, which can be written in form $\widetilde{\mathcal{S}}_k^m$ $(m\geq 3; 2\leq k\leq m-1)$. Let $X=\{x_1,...,x_{m-1}\}$ be one of the sets of variables defining this form. Let $b=\binom{m-1}{k-1}$. Let π be the ordering of variables $y_1,...,y_n$ of embedded functions $\mathcal{F}_1$,...,$\mathcal{F}_b$ with respect to which they are represented by $K_1,...,K_b$ intervals respectively, such that $K=\sum_{i=1}^{b}K_i$ is minimal among all orderings of these variables. Then for any set $X'\neq X$ of $m-1$ variables of function f, which also define its form $\widetilde{\mathcal{S}}_k^m$ holds, that embedded functions $\mathcal{F}'_1,...,\mathcal{F}'_b$ can be represented by $K_1,...,K_b$ intervals respectively with respect to the same ordering of their variables.*

Proof. Due to Lemma 5.4.18, if f has the form $\widetilde{\mathcal{S}}_k^m$, then it can also be written in the form 5.5. Without loss of generality let the set of variables $\{x_{m-1},y_1,...,y_{r-1}\}$ be the set $\{z_1,...,z_r\}$ in the form 5.5. Function f can then have the form $\widetilde{\mathcal{S}}_k^m$ defined by any of the sets $X_i=\{x_1,...,x_{m-2},z_i\}, i\in\{1,...,r\}$ and $X_1=X$.

First, we shall prove, that functions $\mathcal{G}_1, ..., \mathcal{G}_b$ (from 5.5) can be represented by at most $K_1, ..., K_b$ intervals respectively with respect to the same ordering of their variables.

As we already said in Observation 5.4.17, we can get all the functions $\mathcal{F}_l$ each one by fixing $k-1$ variables from X to 1 and the rest to 0. If we do these fixations in our form 5.5 we will get the following.

$$\mathcal{F}_i = \bigwedge_{h=2}^{r} z_h \mathcal{G}_i; \qquad i \in \{1, ..., \tbinom{m-2}{k-2}\}$$
$$\mathcal{F}_j = \bigvee_{h=2}^{r} z_h \vee \mathcal{G}_j; \quad j \in \{\tbinom{m-2}{k-2} + 1, ..., \tbinom{m-1}{k-1}\}$$

Where first $s = \binom{m-2}{k-2}$ fixations are such that $z_1 = x_{m-1}$ is fixed to 1 and in the rest $\binom{m-2}{k-1}$ of them it is fixed to 0.

Due to Corollary 4.3.2 we know, that functions $\mathcal{G}_1 = \mathcal{F}_1[z_2 := 1]...[z_r := 1], ..., \mathcal{G}_s = \mathcal{F}_s[z_2 := 1]...[z_r := 1], \mathcal{G}_{s+1} = \mathcal{F}_{s+1}[z_2 := 0]...[z_r := 0], ..., \mathcal{G}_b = \mathcal{F}_b[z_2 := 0]...[z_r := 0]$ are positive functions represented by $K'_1, ..., K'_r$ intervals respectively with respect to the restriction π_0 of the ordering π to the remaining variables and for each $l \in \{1, ..., b\}$ holds that $K'_l \leq K_l$.

Let us now consider set of variables $X_j, j \neq 1$, w.l.o.g. we choose X_r. The embedded functions for this set are following.

$$\mathcal{F}'_i = \bigwedge_{h=1}^{r-1} z_h \mathcal{G}_i; \qquad i \in \{1, ..., s\}$$
$$\mathcal{F}'_j = \bigvee_{h=1}^{r-1} z_h \vee \mathcal{G}_j; \quad j \in \{s+1, ..., b\}$$

Now according to the Corollary 5.4.3 for each $l \in \{1, ..., b\}$ function $\mathcal{F}_l$ has exactly K'_l intervals with respect to the ordering π_1 of its variables created from ordering π_0 by adding variables $z_1, ..., z_{r-1}$ as the first $r-1$ most significant ones in any ordering.

Moreover if for any l held, that $K'_l < K_l$, then $\sum_{i=1}^{b} K'_i < K$. But that would mean that for ordering π the summation of intervals was not minimal. Therefore $K'_l = K_l$ for each l. ∎

5.5 Recognition of positive 3-interval functions

Lemma 5.5.1. *Let f be a positive 1-interval function on n variables, which is not identically equal to zero. Let π be an ordering, with respect to which f can be represented by one interval, and let x_i be the most significant variable, i.e., $\pi(i) = 1$. Then x_i defines the form $\widetilde{\mathcal{S}}_1^2$ or $\widetilde{\mathcal{S}}_2^2$ of function f.*

This lemma is already introduced and proved in [3] as Corollary 3.8. We will skip the proof and continue with other lemma from [3], which we will prove in a different way. This will prepare us for a more complex proof later.

Lemma 5.5.2. *Let f be a positive Boolean function on n variables which cannot be written in forms $\widetilde{\mathcal{S}}_1^2$ or $\widetilde{\mathcal{S}}_2^2$ (and therefore is not a 1-interval function). Then if f is a 2-interval function, it has form $\widetilde{\mathcal{S}}_2^3$ and embedded functions $\mathcal{F}_1$ and $\mathcal{F}_2$ are 1-interval functions with respect to the same ordering of their variables.*

Proof. In following proof we implicitly use these properties of positive non-constant k-interval function g on m variables and their ordering π:

(i) $g(0) = 0$ and $g(2^n - 1) = 1$ (Lemma 5.1.1)

(ii) if g is 1-interval with respect to the ordering π, $0 \leq a \leq 2^n - 1$ and $g(a) = 0$, then $g(a') = 0$ for all $0 \leq a' \leq a$ according to π (trivial)

(iii) if g is 1-interval with respect to the ordering π, $0 \leq a \leq 2^n - 1$ and $g(b) = 1$, then $g(b') = 1$ for all $b \leq b' \leq 2^n - 1$ according to π (trivial)

(iv) if in arbitrary branching tree of g (with root variable x_i) left half of the tree is constant zero ($g[x_i := 0] = 0$)), then $g = x_i \wedge \mathcal{F}_1$ for certain $\mathcal{F}_1$ thus g has form $\widetilde{\mathcal{S}}_2^2$ (Corollary 5.4.3)

(v) if in arbitrary branching tree of g (with root variable x_i) right half of the tree is constant one ($g[x_i := 1] = 1$)), then $g = x_i \vee \mathcal{F}_1$ for certain $\mathcal{F}_1$ thus g has form $\widetilde{\mathcal{S}}_1^2$ (Corollary 5.4.3)

(vi) $g = g[x_i := 0] \vee x_i g[x_i := 1]$ for any $i \in \{1, ..., m\}$ (Lemma 3.0.4)

Let π be the ordering of variables with respect to which f is represented by 2 intervals. Without loss of generality, π is identity. Of course $f(\boldsymbol{0}) = 0$ and $f(\boldsymbol{1}) = 1$.

We will denote functions on $n - 1$ variables $f_0 = f[x_1 := 0]$ and $f_1 = f[x_1 := 1]$. Since f has neither form $\widetilde{\mathcal{S}}_1^2$ nor $\widetilde{\mathcal{S}}_2^2$, neither one of functions f_0 and f_1 is constant. Hence $f(2^{n-1} - 1) = f(01...1) = f_0(\boldsymbol{1}) = 1$, and $f(2^{n-2}) = f(10...0) = f_1(\boldsymbol{0}) = 0$. We will now explore the branching tree of function f according to the ordering π.

This branching tree represents, of course, 2-interval function. Therefore both f_0 and f_1 are 1-interval with respect to the ordering π restricted on their variables.

Now if $f_1(01...1) = f(101...1) = 0$ (thus $f_1[x_2 := 0]$ being constant 0), then we would from positivity need also $f(001...1) = f_0(01...1) = 0$ and function $f_0[x_2 := 0]$ being constant 0 as well. But that would mean that $f[x_2 := 0]$ would be constant 0 function and therefore f would have form $\widetilde{\mathcal{S}}_2^2$, which is in contradiction with assumptions of the lemma. So $f_1(01...1) = f(101...1) = 1$ and $f_1[x_2 := 1]$ is constant 1, therefore f_1 has form $\widetilde{\mathcal{S}}_1^2$ and $f_1[x_2 := 0]$ is a non-constant 1-interval function with respect to π restricted on its variables. We denote it as $\mathcal{F}_1$.

If $f_0(10...0) = f(010...0) = 1$, then $f_0[x_2 := 1]$ is constant 1, and consequently $f[x_2 := 1]$ constant 1 as well. But this cannot be, because in that case f would have form $\widetilde{\mathcal{S}}_1^2$. Therefore $f_0(10...0) = f(010...0) = 0$ and $f_0[x_2 := 0]$ is constant 0, hence f_0 has form $\widetilde{\mathcal{S}}_2^2$ and $f_0[x_2 := 1]$ is 1-interval function with respect to π restricted on its variables (which are the very same variables as function $f_1[x_2 := 0]$ has). We denote it as $\mathcal{F}_2$. Figure 5.3 captures such positive 2-interval function.

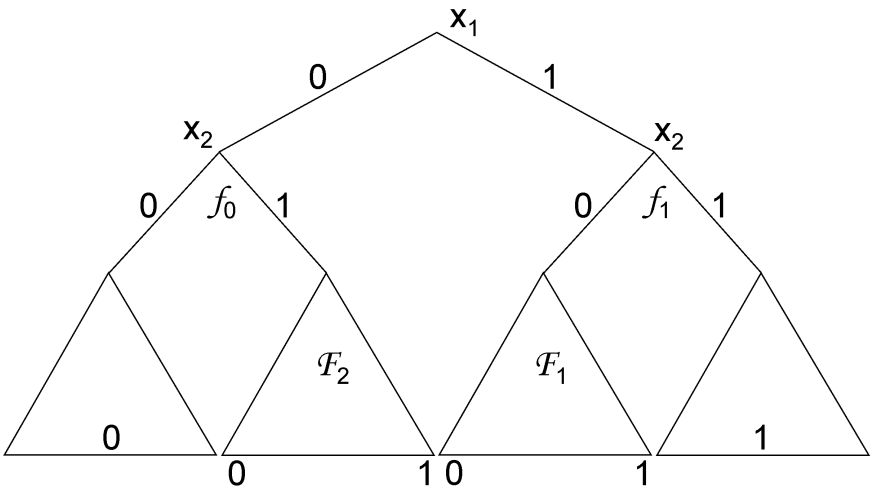


Figure 5.3: Branching tree of a positive 2-interval Boolean function.

If we put this knowledge together, we get that $f = \widetilde{\mathcal{S}}_2^2 \vee x_1\widetilde{\mathcal{S}}_1^2 = x_2\mathcal{F}_2 \vee x_1(x_2 \vee \mathcal{F}_1)$. After applying distributivity we have $f = x_1x_2 \vee x_1\mathcal{F}_1 \vee x_2\mathcal{F}_2$, which is exactly the form $\widetilde{\mathcal{S}}_2^3$. Moreover, functions $\mathcal{F}_1$ and $\mathcal{F}_2$ are positive 1-interval functions with respect to the ordering π restricted on their variables. ∎

Corollary 5.5.3. *Let f be a positive Boolean function on n variables which cannot be written in forms $\widetilde{\mathcal{S}}_1^2$ or $\widetilde{\mathcal{S}}_2^2$. Then $f \in \mathcal{C}_{2-int}^+ \setminus \mathcal{C}_{1-int}^+$ if and only if it*

has form $\widetilde{\mathcal{S}}_2^3$ *and functions* $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *are* 1*-interval functions according to the same ordering of their variables.*

Proof.

(if part) Follows from Lemmas 5.4.9 (there exists ordering π with respect to which function is 2-interval) and 5.4.13 (there does not exist any ordering π' with respect to which function is 1-interval).

(only if part) Is actually Lemma 5.5.2 ∎

Lemma 5.5.4. *Let* f *be a positive Boolean function on* n *variables which cannot be written in forms* $\widetilde{\mathcal{S}}_1^2$ *or* $\widetilde{\mathcal{S}}_2^2$ *(and therefore is not a* 1*-interval function). Then if* f *is* 3*-interval function such that it is not* 2*-interval function, it has one of the following forms:*

(a) $\widetilde{\mathcal{S}}_2^3$ *and one of the functions* $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *is non-constant* 1*-interval function and the other one is* 2*-interval (such that it is not* 1*-interval) function with respect to the same ordering of their variables; also for any ordering of their variables, they cannot be simultaneously represented by one interval.*

(b) $\widetilde{\mathcal{S}}_2^4$ *and* $\mathcal{F}_1$*,* $\mathcal{F}_2$*,* $\mathcal{F}_3$ *are non-constant* 1*-interval functions with respect to the same ordering of their variables.*

(c) $\widetilde{\mathcal{S}}_3^4$ *and* $\mathcal{F}_1$*,* $\mathcal{F}_2$*,* $\mathcal{F}_3$ *are non-constant* 1*-interval functions with respect to the same ordering of their variables.*

(d) 5.6, where functions $\mathcal{F}_1$*,* $\mathcal{F}_2$ *and* $\mathcal{F}_1 \vee \mathcal{F}_2 \vee \mathcal{F}_3$ *are* 1*-interval functions on the same* $n-3$ *variables with respect to the same ordering of their variables;* $\mathcal{F}_1$*,* $\mathcal{F}_2$*,* $\mathcal{F}_3$ *are non-constant functions except* $\mathcal{F}_3$ *can be constant zero*

(e) 5.7, where functions $\mathcal{F}_1$*,* $\mathcal{F}_1 \vee \mathcal{F}_2$ *and* $\mathcal{F}_1 \vee \mathcal{F}_3$ *are* 1*-interval functions on the same* $n-3$ *variables with respect to the same ordering of their variables;* $\mathcal{F}_1$*,* $\mathcal{F}_2$*,* $\mathcal{F}_3$ *cannot be constant one and* $\mathcal{F}_1$ *cannot be either constant zero*

$$x_i x_k \vee x_j x_k \vee x_i \mathcal{F}_1 \vee x_j \mathcal{F}_2 \vee x_i x_j \mathcal{F}_3 \tag{5.6}$$

$$x_i x_j \vee x_k \mathcal{F}_1 \vee x_i x_k \mathcal{F}_2 \vee x_j x_k \mathcal{F}_3 \tag{5.7}$$

Proof. In this proof we implicitly use the very same properties (i) - (vi) of non-constant positive k-interval functions as in proof of the Lemma 5.5.2.

Let π be the ordering of variables with respect to which f is represented by 3 intervals. Without loss of generality, π is identity.

We will again denote $f_0 = f[x_1 := 0]$ and $f_1 = f[x_1 := 1]$. These functions are again not constant. Again $f_0(\boldsymbol{0}) = f_1(\boldsymbol{0}) = 0$ and $f_0(\boldsymbol{1}) = f_1(\boldsymbol{1}) = 1$. Now either f_0 is 1-interval and f_1 is 2-interval, both with respect to π restricted on

their variables, or vice versa. We will explore the branching tree according to the ordering π for both cases.

In following text we will several times find out that our positive 3-interval function f has the form $\widetilde{\mathcal{S}}_2^3$. In all these cases, the two embedded functions cannot be both represented by one interval at once with respect to any ordering of their variables. If they could, function f would be then 2-interval according to the Lemma 5.4.9, which cannot be.

(a) Let f_0 be 1-interval and f_1 2-interval. Figure 5.4 shows, which vectors are considered as we progress on this proof. These vectors are numbered by the order of our progression and they correspond with the numbers in the text.

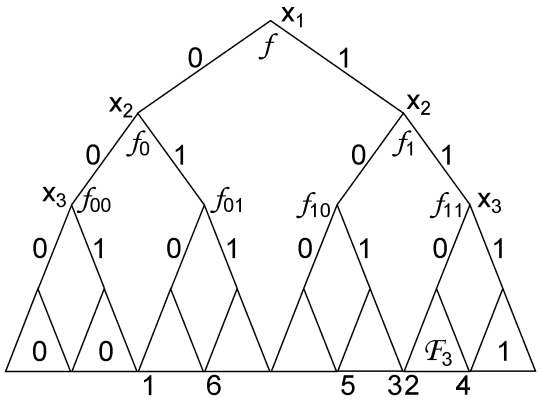


Figure 5.4: Analysis of 3-interval function with 2 intervals in right subtree

(1) If $f_0(10..0) = f(010..0) = 1$, then $f_0[x_2 := 1]$ is constant one and from the positivity $f_1[x_2 := 1]$ is constant one as well. From there $f[x_2 := 1]$ is constant one and f has form $\widetilde{\mathcal{S}}_1^2$ which is against conditions of the lemma. Therefore $f_0(10..0) = f(010..0) = 0$ and $f_0[x_2 := 0]$ is constant zero and f_0 has form $\widetilde{\mathcal{S}}_2^2$.

Now we have two possibilities. **(2)** Either $f_1(10..0) = f(110..0) = 1$ or 0. In the first case $f_1[x_2 := 1]$ is constant one (if it was not, it could not be positive) and f_1 has the form $\widetilde{\mathcal{S}}_1^2$ thus f having form $\widetilde{\mathcal{S}}_2^3$. Specifically $f_1[x_2 := 0] = \mathcal{F}_1$ is 2-interval function, $f_0[x_2 := 1] = \mathcal{F}_1$ is 1-interval function, both with respect to restricted π, and $f = x_2\mathcal{F}_2 \vee x_1(x_2 \vee \mathcal{F}_1) = x_1x_2 \vee x_1\mathcal{F}_1 \vee x_2\mathcal{F}_2$.

We will now move to the second case, where $f_1(10..0) = f(110..0) = 0$. We consider **(3)** $f(101...1) = f_1(01...1) = 0$. In this case $f_1[x_2 := 0]$ would have to be constant zero (otherwise it could not be positive) and since $f_0[x_2 := 0]$ is constant zero, it would hold, that $f[x_2 := 0]$ would be constant zero and therefore f would have form $\widetilde{\mathcal{S}}_2^2$, which cannot be according to our assumptions. Therefore $f_1(01...1) = 1$ and functions $f_0[x_2 := 1]$, $f_1[x_2 := 0]$ and $f_1[x_2 := 1]$ are 1-interval functions with respect to same the ordering π restricted on their variables. We will denote them as f_{01}, f_{10} and f_{11}. Constant zero function $f_0[x_2 := 0]$will be denoted as f_{00}.

(4) We consider $f(1101...1) = f_{11}(01...1) = 0$, thus $f_{11}[x_3 := 0]$ being constant zero. Then from positivity of f we have that $f[1001...1] = f[0101...1] = 0$ and both $f_{01}[x_3 := 0]$ and $f_{10}[x_3 := 0]$ are constant 0. If we put this together with the fact that f_{00} is constant zero as well, we would get that $f[x_3 := 0]$ is constant zero function resulting into f having form $\widetilde{\mathcal{S}}_2^2$, which cannot happen. Therefore $f(1101...1) = f_{11}(01...1) = 1$ and $f_{11}[x_3 := 1]$ is constant one. We will denote non-constant positive 1-interval function $f_{11}[x_3 := 0]$ as $\mathcal{F}_3$.

Next we have two possibilities **(5)** on input vector $(1010...0)$ of function f. Firstly $f(1010...0) = f_{10}(10...0) = 0$ and consequently $f_{10}[x_3 := 0]$ is constant zero, and we denote $f_{10}[x_3 := 1]$ as $\mathcal{F}_2$. This case consists of two more subcases. **(6)** First is $f(0110...0) = f_{01}(10...0) = 0$. Then $f_{01}[x_3 := 0]$ is constant zero and we denote $f_{01}[x_3 := 1]$ as $\mathcal{F}_1$. In this subcase $f = x_2x_3\mathcal{F}_1 \vee x_1(x_3\mathcal{F}_2 \vee x_2(x_3 \vee \mathcal{F}_3)) = x_1x_2x_3 \vee x_2x_3\mathcal{F}_1 \vee x_1x_3\mathcal{F}_2 \vee x_1x_2\mathcal{F}_3$, which is form $\widetilde{\mathcal{S}}_3^4$ with three embedded 1-interval functions.

In the second subcase **(6)** we have $f(0110...0) = f_{01}(10...0) = 1$. Then $f_{01}[x_3 := 1]$ is constant one and we denote 1-interval function $f_{01}[x_3 := 0]$ as $\mathcal{F}_1$ (possibly constant zero). Now we will on our current branching tree apply operator *moveRootDown(3)* and we can see in the Figure 5.5 that we have got a new ordering of variables $\pi_1 = (x_2, x_3, x_1, x_4, ..., x_n)$ with respect to which function f is also 3-interval function. Moreover, we can write $f = x_3(x_1\mathcal{F}_2) \vee x_2(x_3 \vee (\mathcal{F}_1 \vee x_1\mathcal{F}_3)) = x_2x_3 \vee x_2(\mathcal{F}_1 \vee x_1\mathcal{F}_3) \vee x_3(x_1\mathcal{F}_2)$, which is form $\widetilde{\mathcal{S}}_2^3$ with embedded 2-interval function $\mathcal{F}_1' = \mathcal{F}_1 \vee x_1\mathcal{F}_3$ and 1-interval function $\mathcal{F}_2' = x_1\mathcal{F}_2$, both with respect to the ordering π_1 restricted on their variables. Also $\mathcal{F}_1$ cannot be constant zero, otherwise f would be 2-interval function with respect to π_1.

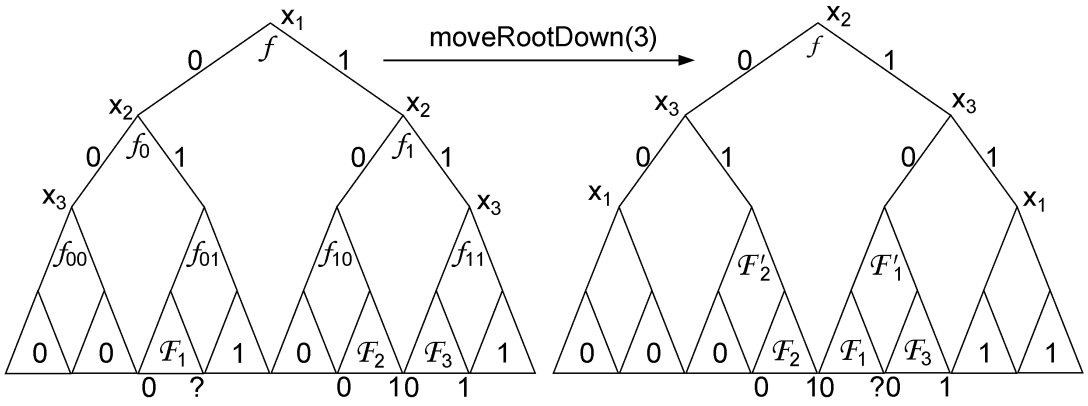


Figure 5.5: One of the cases of positive 3-interval function

Now we proceed with the case, where **(5)** $f(1010...0) = f_{10}(10...0) = 1$ and consequently $f_{10}[x_3 := 1]$ is constant one. We denote 1-interval function $f_{10}[x_3 := 0]$ (possibly constant zero) as $\mathcal{F}_2$.

We will again explore two subcases. **(6)**First is that $f(0110...0) = f_{01}(10...0) = 0$. Then $f_{01}[x_3 := 0]$ is constant zero and we denote 1-interval function $f_{01}[x_3 := 1]$ as $\mathcal{F}_1$. We will now on our branching tree apply operator *moveDown(2,1)* and as we can see in Figure 5.6, we will get a new branching tree of 3-interval function f with respect to the ordering $\pi_2 = (x_1, x_3, x_2, x_4, ..., x_n)$. We can also see that if $\mathcal{F}_2$ was constant zero, function f would be 2-interval with respect to the ordering π_2. Therefore $\mathcal{F}_2$ is not constant zero and we can again write $f = x_3(x_2\mathcal{F}_1) \vee x_1(x_3 \vee (\mathcal{F}_2 \vee x_2\mathcal{F}_3)) = x_1x_3 \vee x_1(\mathcal{F}_2 \vee x_2\mathcal{F}_3) \vee x_3(x_2\mathcal{F}_1)$, which is form $\widetilde{\mathcal{S}}_2^3$ with embedded 2-interval function $\mathcal{F}'_1 = \mathcal{F}_2 \vee x_2\mathcal{F}_3$ and 1-interval function $\mathcal{F}'_2 = x_2\mathcal{F}_1$, both with respect to the ordering π_2 restricted on their variables.

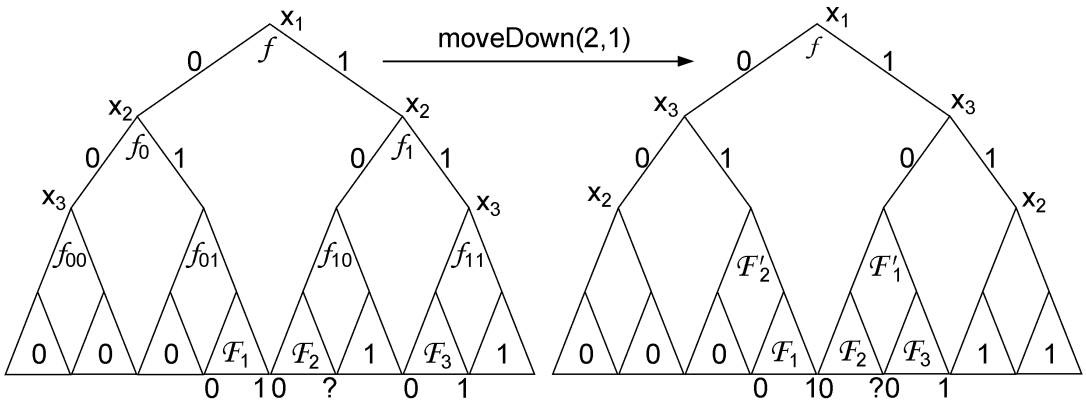


Figure 5.6: One of the cases of positive 3-interval function

(6) The second subcase is that $f(0110...0) = f_{01}(10...0) = 1$. Then $f_{01}[x_3 := 1]$ is constant one and we denote 1-interval function (possibly constant zero) $f_{01}[x_3 := 0]$ as $\mathcal{F}_1$. Now we have $f = x_2(x_3 \vee \mathcal{F}_1) \vee x_1(x_3 \vee \mathcal{F}_2 \vee x_2(x_3 \vee \mathcal{F}_3))$. After applying distributivity and one absorption, we will get $f = x_1x_3 \vee x_2x_3 \vee x_1\mathcal{F}_2 \vee x_2\mathcal{F}_1 \vee x_1x_2\mathcal{F}_3$. From the positivity of f it must hold, that $\mathcal{F}_1 \leq \mathcal{F}_3$ and $\mathcal{F}_2 \leq \mathcal{F}_3$ (i.e. $\mathcal{F}_3$ has the longest truepoint interval), because every input vector of f such that it is also input vector of $\mathcal{F}_3$ (of course restricted on appropriate bits) differs form input vectors corresponding to $\mathcal{F}_1$ in exactly one bit and this bit is 1 for $\mathcal{F}_3$. The reason is the same for the second inequality. Therefore $\mathcal{F}_3$ can be written as $\mathcal{F}_3 = \mathcal{F}_1 \vee \mathcal{F}_2 \vee \mathcal{F}'_3$ and consequently $f = x_1x_3 \vee x_2x_3 \vee x_1\mathcal{F}_2 \vee x_2\mathcal{F}_1 \vee x_1x_2\mathcal{F}_1 \vee x_1x_2\mathcal{F}_1 \vee x_1x_2\mathcal{F}'_3$. If we now again apply distributivity and another several absorptions, we have $f = x_1x_3 \vee x_2x_3 \vee x_1\mathcal{F}_2 \vee x_2\mathcal{F}_1 \vee x_1x_2\mathcal{F}'_3$, which is the form 5.6 such that $\mathcal{F}'_3$ can be constant zero (even though $\mathcal{F}_3$ cannot be constant). Figure 5.7 captures positive 3-interval function which has the form 5.6.

Finally, we have to look what would happen if one of the functions $\mathcal{F}_1$ or $\mathcal{F}_2$ was constant zero. If $\mathcal{F}_1$ was constant zero, we can do the operation *move-*

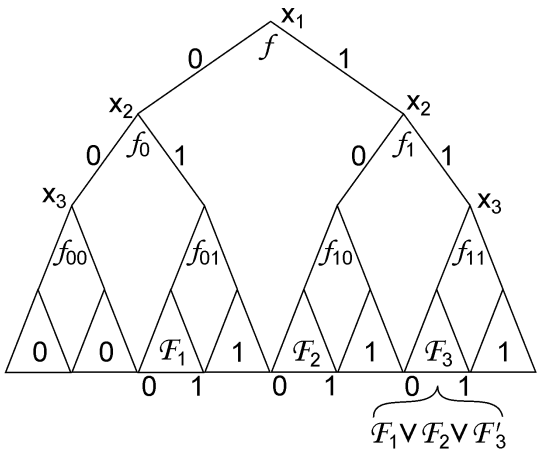


Figure 5.7: Positive 3-interval function having the form 5.6.

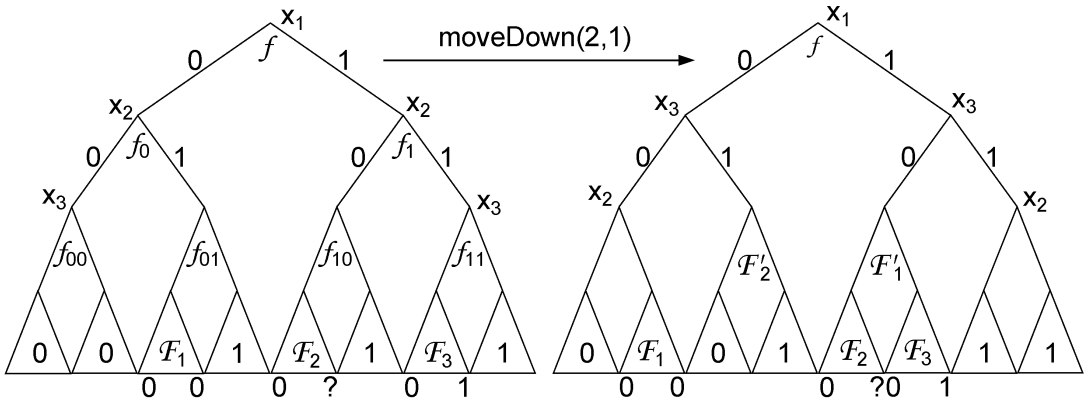


Figure 5.8: One of the cases of positive 3-interval function

Down(2,1) and we will get permutation $\pi_3 = (x_1, x_3, x_2, x_4, ..., x_n)$ with respect to which function f is 3-interval and has form $\widetilde{\mathcal{S}}_2^3 = x_1x_3 \vee x_1(\mathcal{F}_2 \vee x_2\mathcal{F}_3) \vee x_3(x_2)$ as Figure 5.8 shows. Clearly, both of them cannot be constant zero, because in that case f would be 2-interval with respect to the ordering π_3.

If $\mathcal{F}_2$ was constant zero, we can do the operation *moveRootDown(3)* and we will get permutation $\pi_4 = (x_2, x_3, x_1, x_4, ..., x_n)$ with respect to which function f is 3-interval and has form $\widetilde{\mathcal{S}}_2^3 = x_2x_3 \vee x_2(\mathcal{F}_1 \vee x_1\mathcal{F}_3) \vee x_3(x_1)$ as Figure 5.9 shows.

(b) Now let f_0 be 2-interval and f_1 1-interval.

The first several steps of this part of the proof are analogous to the previous part. Using strictly complement thoughts and choices of vectors (e.g. instead of vector (010...0) having the value 1, we will consider vector (101...1) and its value 0) we will get to the following state of the branching tree.

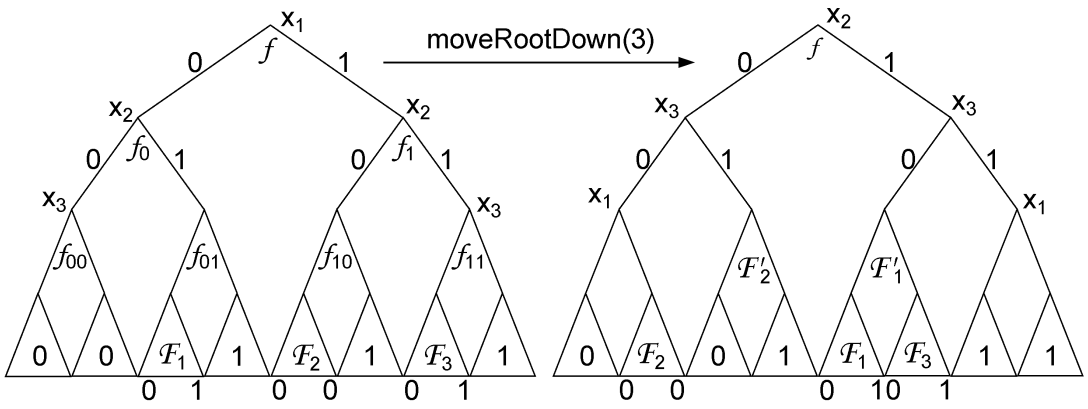


Figure 5.9: One of the cases of positive 3-interval function

$f(0010...0) = f(010...0) = f(10...0) = 0, f(001...1) = f(01...1) = f(101...1) = 1$. We again denote $f_{00}, f_{01}, f_{10}, f_{11}$ the same subtrees as in the previous case and $f_{00}[x_3 := 0]$ is constant zero and f_{11} is constant one. $f_{00}[x_3 := 1], f_{01}, f_{10}$ are positive non-constant 1-interval functions with respect to the ordering π restricted on their variables. $f_{00}[x_3 := 1]$ will be denoted as $\mathcal{F}_1$. Numbers in Figure 5.10 help with better orientation similarly as in the previous case.

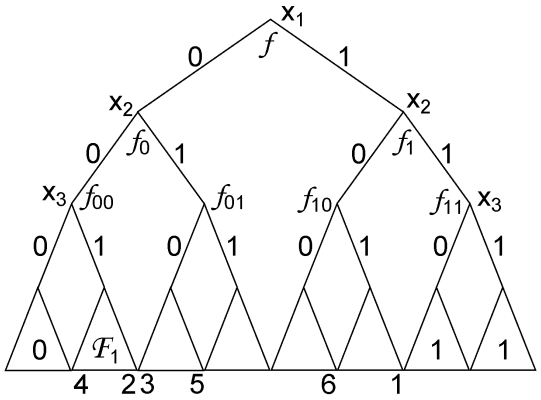


Figure 5.10: Analysis of 3-interval function with 2 intervals in left subtree

Now we will consider **(5)** $f(0101...1) = f_{01}(01...1) = 1$ and consequently $f_{01}[x_3 := 1]$ constant one. We denote $f_{01}[x_3 := 0]$ as $\mathcal{F}_2$. This case consists of two subcases. **(6)** First is $f(1001...1) = f_{10}(01...1) = 1$. Then $f_{10}[x_3 := 1]$ is constant one and we denote $f_{10}[x_3 := 0]$ as $\mathcal{F}_3$. In this subcase $f = x_3\mathcal{F}_1 \vee x_2(x_3 \vee \mathcal{F}_2) \vee x_1(x_2 \vee (x_3 \vee \mathcal{F}_3)) = x_1x_2 \vee x_1x_3 \vee x_2x_3 \vee x_1\mathcal{F}_3 \vee x_2\mathcal{F}_2 \vee x_3\mathcal{F}_1$, which is form $\widetilde{\mathcal{S}}_2^4$ with three embedded 1-interval functions.

In the second subcase **(6)** we have $f(1001...1) = f_{10}(01...1) = 0$. Then $f_{10}[x_3 := 0]$ is constant zero and we denote 1-interval function $f_{10}[x_3 := 1]$

as $\mathcal{F}_3$ (possibly constant one). Now we will on our current branching tree apply the operator *moveRootDown(3)* and we can see in the Figure 5.11 that we have got a new ordering of variables $\pi_5 = (x_2, x_3, x_1, x_4, ..., x_n)$ with respect to which function f is also 3-interval function. Moreover, we can write $f = x_3(\mathcal{F}_1 \vee x_1\mathcal{F}_3) \vee x_2(x_3 \vee (x_1 \vee \mathcal{F}_2)) = x_2x_3 \vee x_2(x_1 \vee \mathcal{F}_2) \vee x_3(\mathcal{F}_1 \vee x_1\mathcal{F}_3)$, which is form $\widetilde{\mathcal{S}}_2^3$ with embedded 2-interval function $\mathcal{F}'_1 = \mathcal{F}_1 \vee x_1\mathcal{F}_3$ and 1-interval function $\mathcal{F}'_2 = x_1 \vee \mathcal{F}_2$, both with respect to the ordering π_5 restricted on their variables. Also $\mathcal{F}_3$ cannot be constant one, otherwise f would be 2-interval function with respect to π_5.

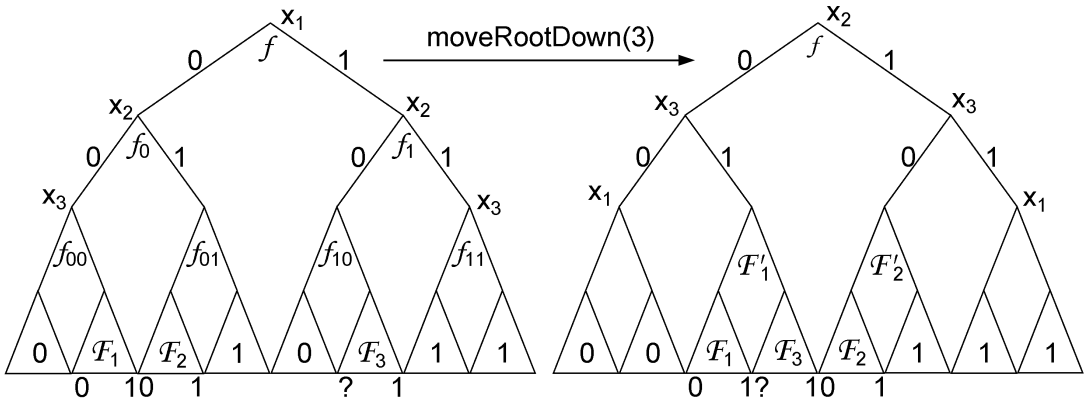


Figure 5.11: One of the cases of positive 3-interval function

Now we proceed with the case, where **(5)** $f(0101...1) = f_{01}(01...1) = 0$ and consequently $f_{01}[x_3 := 0]$ is constant zero. We denote 1-interval function $f_{01}[x_3 := 1]$ (possibly constant one) as $\mathcal{F}_2$.

We will explore another two subcases. **(6)** First is that $f(1001...1) = f_{10}(01...1) = 1$. Then $f_{10}[x_3 := 1]$ is constant one and we denote 1-interval function $f_{10}[x_3 := 0]$ as $\mathcal{F}_3$. We will on this branching tree apply operator *moveDown(2,1)* and as we can see in Figure 5.12, we will get a new branching tree of 3-interval function f with respect to the ordering $\pi_6 = (x_1, x_3, x_2, x_4, ..., x_n)$. We can also see that if $\mathcal{F}_2$ was constant one, function f would be 2-interval with respect to the ordering π_6. Therefore $\mathcal{F}_2$ is not constant one and we can again write $f = x_3(\mathcal{F}_1 \vee x_2\mathcal{F}_2) \vee x_1(x_3 \vee (x_2 \vee \mathcal{F}_3)) = x_1x_3 \vee x_1(x_2 \vee \mathcal{F}_3) \vee x_3(\mathcal{F}_1 \vee x_2\mathcal{F}_2)$, which is form $\widetilde{\mathcal{S}}_2^3$ with embedded 2-interval function $\mathcal{F}'_1 = \mathcal{F}_1 \vee x_2\mathcal{F}_2$ and 1-interval function $\mathcal{F}'_2 = x_2 \vee \mathcal{F}_3$, both with respect to the ordering π_6 restricted on their variables.

The second subcase **(6)** is that $f(1001...1) = f_{10}(01...1) = 0$. Then $f_{10}[x_3 := 0]$ is constant zero and we denote 1-interval function (possibly constant one) $f_{10}[x_3 := 1]$ as $\mathcal{F}_3$. Now we have $f = x_3\mathcal{F}_1 \vee x_2x_3\mathcal{F}_2 \vee (x_3\mathcal{F}_3 \vee x_2) = x_1x_2 \vee x_3\mathcal{F}_1 \vee x_2x_3\mathcal{F}_2 \vee x_1x_3\mathcal{F}_3$. From the positivity of f it must hold, that $\mathcal{F}_1 \leq \mathcal{F}_2$ and $\mathcal{F}_1 \leq \mathcal{F}_3$ (i.e. $\mathcal{F}_1$ has the shortest truepoint interval), because of the same reasons

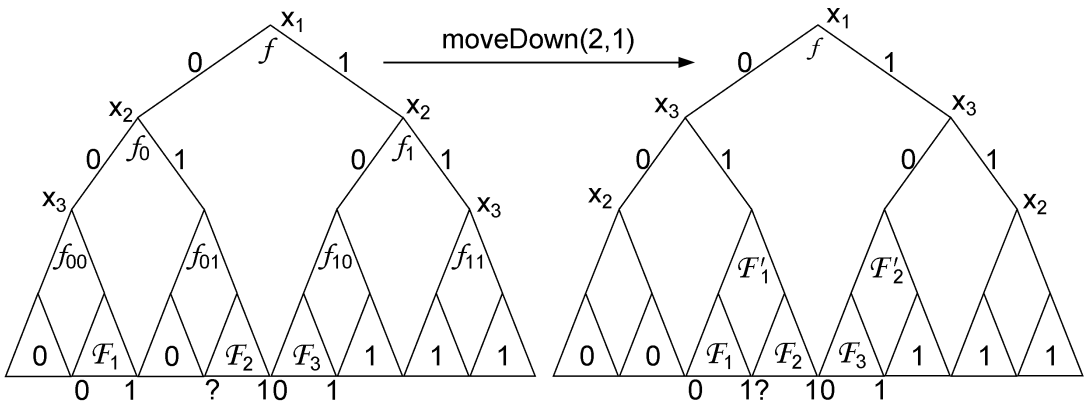


Figure 5.12: One of the cases of positive 3-interval function

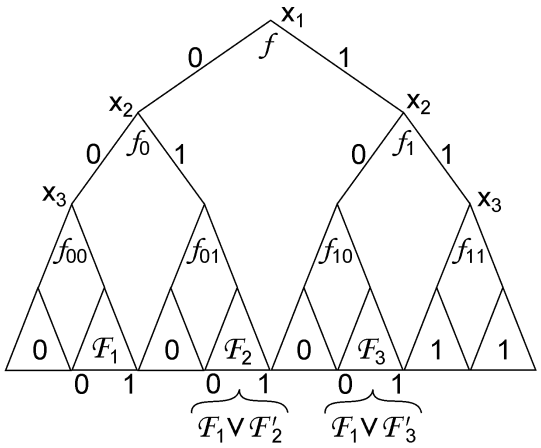


Figure 5.13: Positive 3-interval function having the form 5.7.

as in the analysis of the form 5.6. Therefore $\mathcal{F}_2 = \mathcal{F}_1 \vee \mathcal{F}'_2$ and $\mathcal{F}_3 = \mathcal{F}_1 \vee \mathcal{F}'_3$ and we can write $f = x_1x_2 \vee x_3\mathcal{F}_1 \vee x_2x_3\mathcal{F}_1 \vee x_2x_3\mathcal{F}'_2 \vee x_1x_3\mathcal{F}_1 \vee x_2x_3\mathcal{F}'_3 = x_1x_2 \vee x_3\mathcal{F}_1 \vee x_2x_3\mathcal{F}'_2 \vee x_2x_3\mathcal{F}'_3$, which is form 5.7, such that $\mathcal{F}'_2$ and $\mathcal{F}'_3$ can be constant zero even though $\mathcal{F}_2$ and $\mathcal{F}_3$ cannot. Figure 5.13 shows positive 3-interval function with this form.

At last, we have to show that neither one of the functions $\mathcal{F}_2$ or $\mathcal{F}_3$ cannot be constant one. If $\mathcal{F}_3$ was constant one, we can do the operation *moveDown(2,1)* and we will get permutation $\pi_7 = (x_1, x_3, x_2, x_4, ..., x_n)$ with respect to which function f is 3-interval and has form $\widetilde{\mathcal{S}}_2^3 = x_1x_3 \vee x_1(x_2) \vee x_3(\mathcal{F}_1 \vee x_2\mathcal{F}_2)$ as Figure 5.14 shows. Clearly, both of them cannot be constant one, otherwise f would be 2-interval with respect to the ordering π_7.

If $\mathcal{F}_2$ was constant one, we can do the operation *moveRootDown(3)* and we will get permutation $\pi_8 = (x_2, x_3, x_1, x_4, ..., x_n)$ with respect to which function

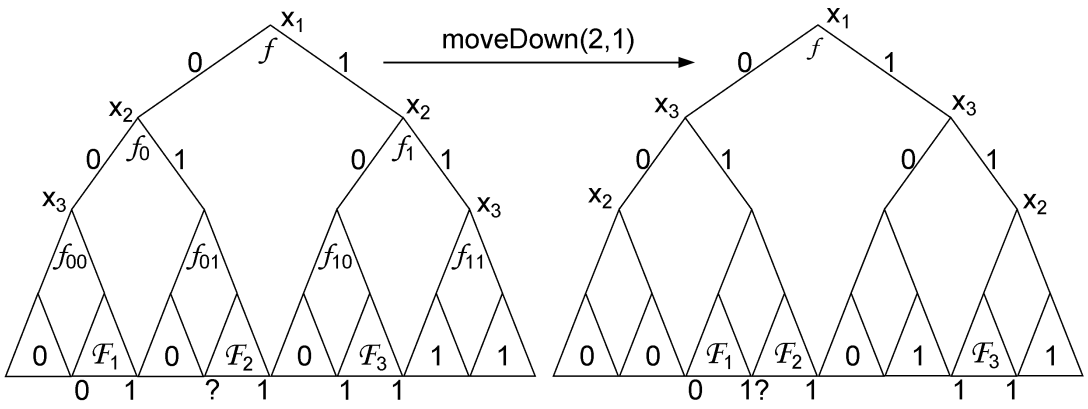


Figure 5.14: One of the cases of positive 3-interval function

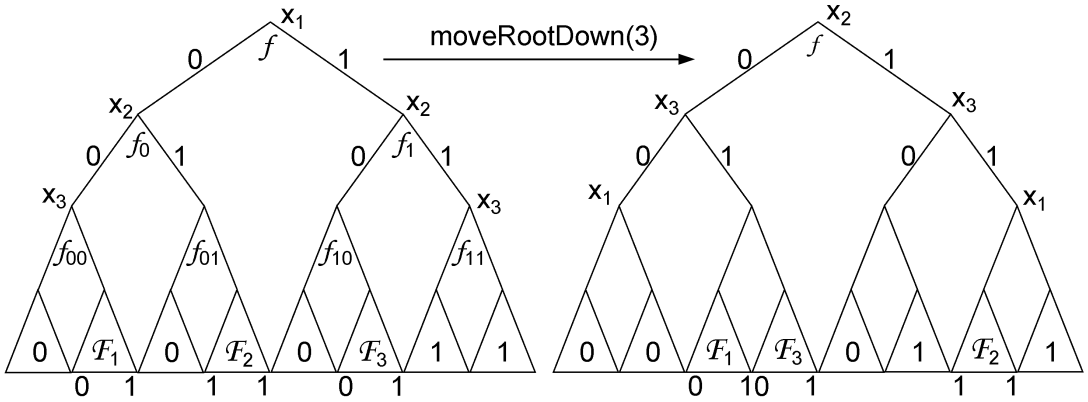


Figure 5.15: The last case of positive 3-interval function

f is 3-interval and has form $\widetilde{\mathcal{S}}_2^3 = x_2x_3 \vee x_2(x_1) \vee x_3(\mathcal{F}_1 \vee x_1\mathcal{F}_3)$ as Figure 5.15 shows.

At this point we have run out of all possible cases of positive function f having three intervals with respect to the ordering π of its variables and the proof is complete. ∎

Observation 5.5.5.
Properties of form 5.6:

(d1) there are two quadratic terms which share one variable (we will call it the *leading variable*) and this variable is not present in any other terms; the rest of the terms contain at least one of the other variables of these quadratic terms (*co-leading variables* x_i, x_j)

(d2) there is at least one term with x_i and one with x_j both without the leading variable

(d3) there are at least four terms

(d4) for each variable holds that it is not present in at least two terms

(d5) there are at least two different pairs of disjoint terms

(d6) there is no term with both co-leading variables of length less than three

Properties of form 5.7:

(e1) there is variable x_k (we will call it the *leading variable*) which is present in every term except of one

(e2) the only term without the leading variable is quadratic (and contains two *co-leading variables*)

(e3) there is no other term with both variables from above mentioned quadratic term (it would be absorbed)

(e4) there is at least one term with the leading variable

(e5) there is at least one pair of disjoint terms

Lemma 5.5.6. *Positive Boolean function f cannot have more than one of the forms (a) - (e) from Lemma 5.5.4 at once.*

Proof. From Lemma 5.4.7 we know, that f cannot have any two of forms $\widetilde{\mathcal{S}}_2^3$, $\widetilde{\mathcal{S}}_2^4$ and $\widetilde{\mathcal{S}}_3^4$ at once. It is also clear that it cannot have form $\widetilde{\mathcal{S}}_3^4$ together with any of the forms 5.6 or 5.7. This is true simply because $\widetilde{\mathcal{S}}_3^4$ has all terms of length at least three and the other two forms contain some quadratic terms. Properties (d2), (d3) and (e1) ensure, that f cannot have forms 5.6 and 5.7 at once as well.

We can observe that there are no disjoint terms in form $\widetilde{\mathcal{S}}_2^3$. Hence it is incompatible with both forms 5.6 and 5.7 (properties (d5) and (e5)).

For each variable in $\widetilde{\mathcal{S}}_2^4$ holds that it is not present in at least three terms, therefore f cannot have this form together with the form 5.7 as well.

In form 5.6 it is not possible to pick set of three variables such that there would be all its 2-element subsets present in it as quadratic terms. Specifically, for leading and co-leading variables it is impossible by their definition and since at least one of co-leading variables is present in each term, there is no other considerable set of variables. Hence f cannot have both forms 5.6 and $\widetilde{\mathcal{S}}_2^4$ simultaneously. ∎

Corollary 5.5.7. *Let f be a positive Boolean function on n variables which cannot be written in forms $\widetilde{\mathcal{S}}_1^2$ or $\widetilde{\mathcal{S}}_2^2$. Then $f \in \mathcal{C}_{3-int}^+ \setminus \mathcal{C}_{2-int}^+$ if and only if it has exactly one the forms (a) - (e) from Lemma 5.5.4.*

Proof. Lemma 5.5.6 ensures, that neither two of *(a) - (e)* can be fulfilled together. Now we proceed with the proof of equivalence.

(only if part) is Lemma 5.5.4.

(if part) for *(a)* From Lemma 5.4.13 we have that $\widetilde{\mathcal{S}}_2^3$ cannot have less than two intervals for any ordering of its variables. Also embedded functions are not 1-interval with respect to the same ordering of their variables, therefore it is not 2-interval function (Lemma 5.5.2) as well. And from Lemma 5.4.16 there exist ordering π of its variables with respect to which $\widetilde{\mathcal{S}}_2^3$ has three intervals.

(if part) for *(b),(c)* Follows from Lemmas 5.4.13 and 5.4.9.

(if part) for *(d),(e)* Figures 5.7 and 5.13 show that functions in these forms can be represented by three intervals and since they cannot have any of the forms $\widetilde{\mathcal{S}}_1^2$, $\widetilde{\mathcal{S}}_2^2$ and $\widetilde{\mathcal{S}}_2^3$, they cannot be represented by less intervals (Lemmas 5.2.1 and 5.5.2). ∎

We already know, how to recognize positive 1-interval function. Corollary 5.5.3 gives us the hint, how to recognize positive 2-interval function, which is not 1-interval. We can look for the suitable pair of variables, which define its form $\widetilde{\mathcal{S}}_2^3$, such that its embedded functions are 1-interval. Lemma 5.4.19 ensures us that if any such pair of variables exists, then all pairs of variables defining other forms $\widetilde{\mathcal{S}}_2^3$ have also embedded 1-interval functions. Therefore we can choose any one of them.

The recognition algorithm for positive 2-interval functions was already introduced in [4], but the proof of correctness was too specific for that problem. We used more general lemmas, which can be reused also for positive 3-interval functions and even for some cases of k-interval functions, where $k \geq 4$.

Now, as Corollary 5.5.7 suggests, if we have a positive function, which is 3-interval and not 2-interval, we can look for forms *(a) - (e)* from Lemma 5.5.4. If we find pair or triplet of variables defining forms $\widetilde{\mathcal{S}}_2^3, \widetilde{\mathcal{S}}_2^4, \widetilde{\mathcal{S}}_3^4$, Lemma 5.4.19 again ensures us that we do not have to consider any other ones. However, if we find leading and co-leading variables for forms 5.6 or 5.7, we do not know if we have the suitable ones, if we really have the ones for which the embedded functions are 1-interval with respect to the same ordering of their variables. Following lemmas will solve this problem and allow us to pick any such triplet of variables. We will be then almost ready to construct the recognition algorithm.

Lemma 5.5.8. *Let f be a positive 3-interval Boolean function which has the form (d) from Lemma 5.5.4 with the leading variable x_k and co-leading variables x_i, x_j. Then for any other leading triplet of variables x'_i, x'_j, x'_k which define another form 5.6 with embedded functions $\mathcal{F}'_1, \mathcal{F}'_2, \mathcal{F}'_3$ holds, that functions $\mathcal{F}'_1, \mathcal{F}'_2$ and $\mathcal{F}'_1 \vee \mathcal{F}'_2 \vee \mathcal{F}'_3$ are 1-interval with respect to the same ordering of their variables.*

Proof. First we consider the most basic function f which has the form 5.6. In function $x_1x_3 \vee x_2x_3 \vee x_1x_4 \vee x_2x_4$ can obviously be any variable chosen as the

leading one, consequently define co-leading variables and for each such choice the lemma holds.

Now we consider a more complex function $f = x_i x_k \vee x_j x_k \vee x_i \mathcal{F}_1 \vee x_j(\mathcal{F}_2 \vee x_i \mathcal{F}_3)$. Obviously, because of the property (d1), it is not possible to find triplet of variables disjoint with set $\{x_i, x_j, x_k\}$.

First case is that one of the co-leading variables can be also leading variable. Let it be x_i. This means that x_k becomes co-leading variable and $\mathcal{F}_1$ is function of one variable x_l and this variable becomes the second co-leading variable. Also $\mathcal{F}_3$ is constant zero, otherwise x_i would be in more than two terms. And since $x_j \mathcal{F}_2$ does not contain variable x_i, it must be possible to write $\mathcal{F}_2 = x_l \mathcal{F}$, where $\mathcal{F}$ is not constant one (otherwise we would get the most basic case) and does not contain any of variables x_i, x_j, x_k, x_l. Consequently $f = x_k x_i \vee x_l x_i \vee x_k x_j \vee x_l x_j \mathcal{F}$ and $\mathcal{F}_1 = x_l$, $\mathcal{F}'_1 = x_j$, $\mathcal{F}'_2 = x_j \mathcal{F}$, $\mathcal{F}'_3 = 0$. Furthermore, there is no other triplet of leading variables because we have only three quadratic terms with two possible leading variables and the rest of the terms are of length at least three.

And since functions $\mathcal{F}_1 = x_l$ and $\mathcal{F}_2 = \mathcal{F}_1 \vee \mathcal{F}_2 \vee \mathcal{F}_3 = x_l \mathcal{F}$ are 1-interval with respect to the same ordering π of their variables, from Corollary 4.3.2 we have that if we fix x_l appropriately, $\mathcal{F}$ is 1-interval with respect to the ordering π_1 created from π by restriction to the remaining variables (second function will become constant, which is 1-interval with respect to any ordering). Finally, when we apply Corollary 5.4.3, we have that functions $\mathcal{F}'_1$ and $\mathcal{F}'_2 = \mathcal{F}'_1 \vee \mathcal{F}'_2 \vee \mathcal{F}'_3$ are 1-interval with respect to the ordering π_2 created from π_1 by adding x_j as the most significant variable.

If we consider case, in which the leading or any other variable becomes co-leading, we will get again the previous case. Therefore, in all other cases co-leading variables remain the same and some other variable becomes the leading one. We can than write the function f as follows.

$$x_i(\bigvee_{r=1}^{l} x_r \vee \mathcal{G}_1) \vee x_j(\bigvee_{r=1}^{l} x_r \vee \mathcal{G}_2) \vee x_i x_j \mathcal{G}_3$$

Where all variables x_r can be the leading ones and x_k is one of them. The rest of the proof is analogous to the proof of Lemma 5.4.19. Using the Corollary 4.3.2, we will find out that functions $\mathcal{G}_1, \mathcal{G}_2$ and $\mathcal{G}_1 \vee \mathcal{G}_2 \vee \mathcal{G}_3$ are 1-interval with respect to the same ordering π_3 derived from π by restriction to their variables. Finally, using Corollary 5.4.3, we will be able to see that for any $x_q \in \{x_1, ..., x_r\}$ considered as the leading variable, functions $\mathcal{F}'_1, \mathcal{F}'_2$ and $\mathcal{F}'_1 \vee \mathcal{F}'_2 \vee \mathcal{F}'_3$ are 1-interval with respect to the ordering π_4 created from π_3 by adding variables $x_1, ..., x_{q-1}, x_{q+1}, ..., x_r$ as the most significant ones in any ordering. ∎

Lemma 5.5.9. *Let f be a positive 3-interval Boolean function which has the form (e) from Lemma 5.5.4 with the leading variable x_k and co-leading variables x_i, x_j. Then for any other leading triplet of variables x'_i, x'_j, x'_k which define another*

form 5.7 with embedded functions $\mathcal{F}'_1, \mathcal{F}'_2, \mathcal{F}'_3$ holds, that functions $\mathcal{F}'_1, \mathcal{F}'_1 \vee \mathcal{F}'_2$ and $\mathcal{F}'_1 \vee \mathcal{F}'_3$ are 1-interval with respect to the same ordering of their variables.

Proof. First we consider the most basic function f which has the form 5.7. In function $x_1x_2 \vee x_3x_4$ can obviously be any variable chosen as the leading one, consequently define co-leading variables and for each such choice the lemma holds.

Now we consider a more complex function $f = x_ix_j \vee x_k\mathcal{F}_1 \vee x_kx_i\mathcal{F}_2 \vee x_kx_j\mathcal{F}_3$. Obviously, because of the properties (e1) and (e2), it is not possible to find triplet of variables disjoint with set $\{x_i, x_j, x_k\}$.

If one of the co-leading variables, w.l.o.g. x_i, becomes leading variable, then $\mathcal{F}_3$ must be constant zero. And since x_k is present in each one of the rest of the terms, it must become the co-leading variable. Then $\mathcal{F}_1 = x_l$ and x_l is the second co-leading variable and we can write $f = x_kx_l \vee x_ix_j \vee x_ix_k\mathcal{F}$ and $\mathcal{F}=\mathcal{F}_2$ cannot be constant zero (otherwise we would get the most basic case). Also $\mathcal{F}'_1 = x_j, \mathcal{F}'_2 = \mathcal{F}$ and $\mathcal{F}'_3 = 0$. Furthermore, there is no other triplet of leading variables because we have only two disjoint quadratic terms with two possible pairs of co-leading variables and all other terms are of length at least three.

Similarly as in previous proof, since functions $\mathcal{F}_1 = \mathcal{F}_1 \vee \mathcal{F}_3 = x_l$ and $\mathcal{F}_1 \vee \mathcal{F}_2 = x_l \vee \mathcal{F}$ are 1-interval with respect to the same ordering π of their variables, from Corollary 4.3.2 we have that $\mathcal{F}$ is 1-interval with respect to the ordering π_1 created from π by restriction to the remaining variables. Finally, when we apply Corollary 5.4.3 once again we have that functions $\mathcal{F}'_1 = \mathcal{F}'_1 \vee \mathcal{F}'_3$ and $\mathcal{F}'_1 \vee \mathcal{F}'_2$ are 1-interval with respect to the ordering π_2 created from π_1 by adding x_j as the most significant variable.

If we consider case, where the leading or any other variable becomes co-leading, we will again get the previous case. Therefore, in all other cases co-leading variables remain the same and some other variable becomes the leading one. We can than write the function f as follows.

$$x_ix_j \vee (\bigwedge_{r=1}^{l} x_r \wedge \mathcal{G}_1) \vee (\bigwedge_{r=1}^{l} x_r \wedge x_i\mathcal{G}_2) \vee (\bigwedge_{r=1}^{l} x_r \wedge x_j\mathcal{G}_3)$$

Where all variables x_r can be the leading ones, x_k is one of them and the rest of the proof is analogous to the proof of the previous lemma. ∎

Before we introduce the recognition algorithm, we have to deal with one more problem. Specifically, we need to analyze the form $\widetilde{\mathcal{S}}_2^3$ of positive 3-interval function f which is not 2-interval. This is the case when for embedded functions $\mathcal{F}_1$ and $\mathcal{F}_2$ does not exist ordering π with respect to which they are both 1-interval.

As algorithm for recognition of positive 1-interval function presents, at each step there is some non-empty set I (if it is empty, function is not 1-interval) of

variables, from which we can choose next variable for the constructed ordering. It is always the set of all variables constituting linear terms or the set of all variables contained in all terms. It is clear, that we can always choose any one of them. If two (or more) functions have shared ordering of variables, with respect to which they are 1-interval, the intersection I_0 of their sets I_1, I_2 has to be always non-empty.

If the shared ordering does not exist, I_0 has to be at some point (sooner then we process all variables) empty. This can happen in following ways:

1 The set I_i for exactly one of the functions becomes empty.

2 The sets I_1, I_2 for both functions become empty.

3 Both sets I_1, I_2 are non-empty but they are disjoint.

Let us analyze the first case, when the set I_i became empty for exactly one of the functions (let it be $\mathcal{F}_2$ with set I_2). It means, that this function has neither linear term nor one common variable in all terms. Therefore, due to Lemma 5.5.1, it is not 1-interval and consequently it must be 2-interval and have the form $\widetilde{\mathcal{S}}_2^3$ (otherwise the function f is not 3-interval).

The second case is not interesting for us, because it means, that neither one of the embedded functions is 1-interval and therefore f cannot be 3-interval.

The last case is the most complicated. It could mean two more scenarios. The first is that both functions are 1-interval, only with respect to different orderings. The second is that one of them is 1-interval and the other one is 2-interval, but it has the form $\widetilde{\mathcal{S}}_1^2$ or $\widetilde{\mathcal{S}}_2^2$.

We will deal with all this cases in our implementation, which will be described in the proof of correctness and time complexity of our algorithm. For that we need the last lemma.

Lemma 5.5.10. *Let the positive non-constant function f on n variables have exactly $1 \leq m < n$ variables present as linear terms or present in all its terms. Then if we fix any other variable, we will get non-constant function.*

Proof.

(a) Without loss of generality, let $x_1, ..., x_m$ be all variables present as linear terms. Then $M = \{x_{m+1}, ..., x_n\}$ is non-empty set of remaining variables. It is clear, that if we fix any variable from M to zero, then all variables $x_1, ..., x_m$ remain present as linear terms and function is not constant. From positivity if for any $j \in \{1, ..., n\}$ held that $f[x_j := 1] = 0$, then also $f[x_j := 0] = 0$ and f would have to be constant zero, which is forbidden by our assumptions. If for any $x_j \in M$ held that $f[x_j := 1] = 1$, then from Lemma 5.4.2 x_j is also present as linear term and f would not contain exactly m linear terms but more.

(b) The proof is analogous for variables occurring in all terms. ∎

Algorithm 2 - Positive 3-Interval Function Recognition

Input: A nonempty prime and irredundant positive DNF $\mathcal{F}$ on n variables representing function f.

Output: Order $x_1, ..., x_n$ of variables and n-bit numbers a^1, b^1, a^2, b^2, a^3 if $\mathcal{F}$ represents a positive 3-interval function with respect to this order defined by interval $[a^1, b^1], [a^2, b^2]$ and $[a^3, 2^n - 1]$. **NO** otherwise.

if $\mathcal{F}(\mathbf{1}) = 0$ **then** $\mathcal{F}$ is constant 0, output empty interval **halt endif**
if $\mathcal{F}(\mathbf{0}) = 1$ **then** $\mathcal{F}$ is constant 1, tautology, output $[0, 2^n - 1]$ **halt endif**
$a^1 := 0$ $\quad \#c$ will denote the actual value of $a - 1$
$i := 1$
$\mathcal{F}_1 := \mathcal{F}$
while $\mathcal{F}_i \neq \emptyset$ **and** $i \leq n$ **and** $\exists j(\mathcal{F}_i[z_j := 0] = 0 \vee \mathcal{F}_i[x_j := 1] = 1)$ **do**
    if $\mathcal{F}_i[z_j := 0] = 0$ **then** $\quad \#x_j$ appears in every term
        $a_i^1 := 1$
        $\mathcal{F}_{i+1} := \mathcal{F}_i[z_j := 1]$
    else $\quad \#x_j$ forms a linear term
        $\mathcal{F}_{i+1} := \mathcal{F}_i[z_j := 1]$
    endif
    $\pi(i) := j$
    $i := i + 1$
done
if $\mathcal{F} = \emptyset$ **then** $\quad$ #in this case f can be represented by one interval
    while $i \leq n$ **do**
        $\pi(j) := i$, for some $j \leq n$, which is not mapped yet
        $i := i + 1$
    done
    output ordering π and interval $[a^1, 2^n - 1]$
    halt
endif
$b^1 := a^2 := a^1$
if $\exists j, k$ such that x_j, x_k define form $\widetilde{\mathcal{S}}_2^3$ of $\mathcal{F}_i$ **then**
    if $\mathcal{F}_i[x_j := 0][x_k := 1] \in \mathcal{C}^+_{1-int} \wedge \mathcal{F}_i[x_j := 1][x_k := 0] \in \mathcal{C}^+_{1-int}$ and both with respect to the same ordering π' **then**
        $\pi(i) := j;\ \pi(i+1) := k$
        $a_i^1 := 0;\ a_{i+1}^1 := 1$
        $b_i^1 := 0;\ b_{i+1}^1 := 1$
        $a_i^2 := 1;\ a_{i+1}^2 := 0$
        Find interval $[c^1, 2^{n-i-1} - 1]$ representing $\mathcal{F}_i[x_j := 0][x_k := 1]$ with respect to ordering π'
        Find interval $[c^2, 2^{n-i-1} - 1]$ representing $\mathcal{F}_i[x_j := 1][x_k := 0]$ with respect to ordering π'
        $a^1 := a_1^1 ... a_{i+1}^1 c^1$

$b^1 := b^1_1 ... b^1_{i+1} 11...1$
$a^2 := a^2_1 ... a^2_{i+1} c^2$
for all $k \leq n$ unmapped by π **do**
$\pi(k) = \pi'(k - i - 1)$
done
output ordering π and intervals $[a^1, b_1]$ and $[a^2, 2^n - 1]$
halt
endif
$b^2 := a^3 := a^2$
if $\mathcal{F}_i[x_j := 0][x_k := 1] \in \mathcal{C}^+_{1-int} \wedge \mathcal{F}_i[x_j := 1][x_k := 0] \in \mathcal{C}^+_{2-int}$ and both with respect to the same ordering π' **then**
$\pi(i) := j;\ \pi(i+1) := k$
$a^1_i := 0;\ a^1_{i+1} := 1$
$b^1_i := 0;\ b^1_{i+1} := 1$
$a^2_i := 1;\ a^2_{i+1} := 0$
$b^2_i := 1;\ b^1_{i+1} := 0$
$a^3_i := 1;\ a^2_{i+1} := 0$
Find interval $[c^1, 2^{n-i-1} - 1]$ representing $\mathcal{F}_i[x_j := 0][x_k := 1]$ with respect to ordering π'
Find intervals $[c^2, d^2], [c^3, 2^{n-i-1} - 1]$ representing $\mathcal{F}_i[x_j := 1][x_k := 0]$ with respect to ordering π'
$a^1 := a^1_1 ... a^1_{i+1} c^1$
$b^1 := b^1_1 ... b^1_{i+1} 11...1$
$a^2 := a^2_1 ... a^2_{i+1} c^2$
$b^2 := b^2_1 ... b^2_{i+1} d^2$
$a^3 := a^3_1 ... a^3_{i+1} c^3$
for all $k \leq n$ unmapped by π **do**
$\pi(k) = \pi'(k - i - 1)$
done
output ordering π and intervals $[a^1, b_1], [a^2, b_2]$ and $[a^3, 2^n - 1]$
halt
elseif $\mathcal{F}_i[x_j := 0][x_k := 1] \in \mathcal{C}^+_{2-int} \wedge \mathcal{F}_i[x_j := 1][x_k := 0] \in \mathcal{C}^+_{1-int}$ and both with respect to the same ordering π' **then**
$\pi(i) := j;\ \pi(i+1) := k$
$a^1_i := 0;\ a^1_{i+1} := 1$
$b^1_i := 0;\ b^1_{i+1} := 1$
$a^2_i := 0;\ a^2_{i+1} := 1$
$b^2_i := 0;\ b^1_{i+1} := 1$
$a^3_i := 1;\ a^2_{i+1} := 0$
Find intervals $[c^1, d^1], [c^2, 2^{n-i-1} - 1]$ representing $\mathcal{F}_i[x_j := 0][x_k := 1]$ with respect to ordering π'
Find interval $[c^3, 2^{n-i-1} - 1]$ representing $\mathcal{F}_i[x_j := 1][x_k := 0]$ with respect to ordering π'

$a^1 := a^1_1 ... a^1_{i+1} c^1$
$b^1 := b^1_1 ... b^1_{i+1} d^1$
$a^2 := a^2_1 ... a^2_{i+1} c^2$
$b^2 := b^2_1 ... b^2_{i+1} 11..1$
$a^3 := a^3_1 ... a^3_{i+1} c^3$
for all $k \leq n$ unmapped by π **do**
$\pi(k) = \pi'(k-i-1)$
done
output ordering π and intervals $[a^1, b_1], [a^2, b_2]$ and $[a^3, 2^n - 1]$
halt
endif
endif
$b^2 := a^3 := a^2$
if $\exists j, k, l$ (x_j, x_k, x_l define form $\widetilde{\mathcal{S}}^4_2$ of $\mathcal{F}_i$ and $\mathcal{F}_i[x_j := 0][x_k := 0][x_l := 1] \in \mathcal{C}^+_{1-int}$ $\wedge\ \mathcal{F}_i[x_j := 0][x_k := 1][x_l := 0] \in \mathcal{C}^+_{1-int} \wedge \mathcal{F}_i[x_j := 1][x_k := 0][x_l := 0] \in \mathcal{C}^+_{1-int}$ all with respect to the same ordering π') **then**
$\pi(i) := j;\ \pi(i+1) := k;\ \pi(i+2) := l$
$a^1_i := 0;\ a^1_{i+1} := 0;\ a^1_{i+2} := 1$
$b^1_i := 0;\ b^1_{i+1} := 0;\ b^1_{i+2} := 1$
$a^2_i := 0;\ a^2_{i+1} := 1;\ a^2_{i+2} := 0$
$b^2_i := 0;\ b^2_{i+1} := 1;\ b^2_{i+2} := 0$
$a^3_i := 1;\ a^3_{i+1} := 0;\ a^3_{i+2} := 0$
Find interval $[c^1, 2^{n-i-2} - 1]$ representing $\mathcal{F}_i[x_j := 0][x_k := 0][x_l := 1]$ with respect to ordering π'
Find interval $[c^2, 2^{n-i-2} - 1]$ representing $\mathcal{F}_i[x_j := 0][x_k := 1][x_l := 0]$ with respect to ordering π'
Find interval $[c^3, 2^{n-i-2} - 1]$ representing $\mathcal{F}_i[x_j := 1][x_k := 0][x_l := 0]$ with respect to ordering π'
$a^1 := a^1_1 ... a^1_{i+2} c^1$
$b^1 := b^1_1 ... b^1_{i+2} 11...1$
$a^2 := a^2_1 ... a^2_{i+2} c^2$
$b^2 := b^2_1 ... b^2_{i+2} 11...1$
$a^3 := a^3_1 ... a^3_{i+2} c^3$
for all $k \leq n$ unmapped by π **do**
$\pi(k) = \pi'(k-i-2)$
done
output ordering π and intervals $[a^1, b^1], [a^2, b^2]$ and $[a^3, 2^n - 1]$
halt
elseif $\exists j, k, l$ (x_j, x_k, x_l define form $\widetilde{\mathcal{S}}^4_3$ of $\mathcal{F}_i$ and $\mathcal{F}_i[x_j := 0][x_k := 1][x_l := 1] \in \mathcal{C}^+_{1-int} \wedge\ \mathcal{F}_i[x_j := 1][x_k := 0][x_l := 1] \in \mathcal{C}^+_{1-int} \wedge \mathcal{F}_i[x_j := 1][x_k := 1][x_l := 0] \in \mathcal{C}^+_{1-int}$ all with respect to the same ordering π') **then**
$\pi(i) := j;\ \pi(i+1) := k;\ \pi(i+2) := l$
$a^1_i := 0;\ a^1_{i+1} := 1;\ a^1_{i+2} := 1$

$b_i^1 := 0;\ b_{i+1}^1 := 1;\ b_{i+2}^1 := 1$

$a_i^2 := 1;\ a_{i+1}^2 := 0;\ a_{i+2}^2 := 1$

$b_i^2 := 1;\ b_{i+1}^2 := 0;\ b_{i+2}^2 := 1$

$a_i^3 := 1;\ a_{i+1}^3 := 1;\ a_{i+2}^3 := 0$

Find interval $[c^1, 2^{n-i-2} - 1]$ representing $\mathcal{F}_i[x_j := 0][x_k := 1][x_l := 1]$ with respect to ordering π'

Find interval $[c^2, 2^{n-i-2} - 1]$ representing $\mathcal{F}_i[x_j := 1][x_k := 0][x_l := 1]$ with respect to ordering π'

Find interval $[c^3, 2^{n-i-2} - 1]$ representing $\mathcal{F}_i[x_j := 1][x_k := 1][x_l := 0]$ with respect to ordering π'

$a^1 := a_1^1 ... a_{i+2}^1 c^1$

$b^1 := b_1^1 ... b_{i+2}^1 11...1$

$a^2 := a_1^2 ... a_{i+2}^2 c^2$

$b^2 := b_1^2 ... b_{i+2}^2 11...1$

$a^3 := a_1^3 ... a_{i+2}^3 c^3$

for all $k \leq n$ unmapped by π **do**

    $\pi(k) = \pi'(k - i - 2)$

done

output ordering π and intervals $[a^1, b^1], [a^2, b^2]$ and $[a^3, 2^n - 1]$

halt

elseif $\exists j, k, l$ (x_j, x_k, x_l define form 5.6 of $\mathcal{F}_i$ with leading variable x_l and $\mathcal{F}_i[x_j := 0][x_k := 1][x_l := 0] \in \mathcal{C}_{1-int}^+ \wedge\ \mathcal{F}_i[x_j := 1][x_k := 0][x_l := 0] \in \mathcal{C}_{1-int}^+ \wedge \mathcal{F}_i[x_j := 1][x_k := 1][x_l := 0] \in \mathcal{C}_{1-int}^+$ all with respect to the same ordering π')

then

$\pi(i) := j;\ \pi(i+1) := k;\ \pi(i+2) := l$

$a_i^1 := 0;\ a_{i+1}^1 := 1;\ a_{i+2}^1 := 0$

$b_i^1 := 0;\ b_{i+1}^1 := 1;\ b_{i+2}^1 := 0$

$a_i^2 := 1;\ a_{i+1}^2 := 0;\ a_{i+2}^2 := 0$

$b_i^2 := 1;\ b_{i+1}^2 := 0;\ b_{i+2}^2 := 0$

$a_i^3 := 1;\ a_{i+1}^3 := 1;\ a_{i+2}^3 := 0$

Find interval $[c^1, 2^{n-i-2} - 1]$ representing $\mathcal{F}_i[x_j := 0][x_k := 1][x_l := 0]$ with respect to ordering π'

Find interval $[c^2, 2^{n-i-2} - 1]$ representing $\mathcal{F}_i[x_j := 1][x_k := 0][x_l := 0]$ with respect to ordering π'

Find interval $[c^3, 2^{n-i-2} - 1]$ representing $\mathcal{F}_i[x_j := 1][x_k := 1][x_l := 0]$ with respect to ordering π'

$a^1 := a_1^1 ... a_{i+2}^1 c^1$

$b^1 := b_1^1 ... b_{i+2}^1 11...1$

$a^2 := a_1^2 ... a_{i+2}^2 c^2$

$b^2 := b_1^2 ... b_{i+2}^2 11...1$

$a^3 := a_1^3 ... a_{i+2}^3 c^3$

for all $k \leq n$ unmapped by π **do**

    $\pi(k) = \pi'(k - i - 2)$

166 **done**
167 **output** ordering π and intervals $[a^1, b^1], [a^2, b^2]$ and $[a^3, 2^n - 1]$
168 **halt**
169 **elseif** $\exists j, k, l$ (x_j, x_k, x_l define form 5.7 of $\mathcal{F}_i$ with leading variable x_l and
170 $\mathcal{F}_i[x_j := 0][x_k := 1][x_l := 0] \in \mathcal{C}^+_{1-int} \wedge \mathcal{F}_i[x_j := 1][x_k := 0][x_l := 0] \in \mathcal{C}^+_{1-int}$
171 $\wedge\ \mathcal{F}_i[x_j := 1][x_k := 1][x_l := 0] \in \mathcal{C}^+_{1-int}$ all with respect to the same ordering π')
172 **then**
173 $\pi(i) := j;\ \pi(i+1) := k;\ \pi(i+2) := l$
174 $a^1_i := 0;\ a^1_{i+1} := 0;\ a^1_{i+2} := 1$
175 $b^1_i := 0;\ b^1_{i+1} := 0;\ b^1_{i+2} := 1$
176 $a^2_i := 0;\ a^2_{i+1} := 1;\ a^2_{i+2} := 1$
177 $b^2_i := 0;\ b^2_{i+1} := 1;\ b^2_{i+2} := 1$
178 $a^3_i := 1;\ a^3_{i+1} := 0;\ a^3_{i+2} := 1$
179 Find interval $[c^1, 2^{n-i-2} - 1]$ representing $\mathcal{F}_i[x_j := 0][x_k := 0][x_l := 1]$ with
180 respect to ordering π'
181 Find interval $[c^2, 2^{n-i-2} - 1]$ representing $\mathcal{F}_i[x_j := 0][x_k := 1][x_l := 1]$ with
182 respect to ordering π'
183 Find interval $[c^3, 2^{n-i-2} - 1]$ representing $\mathcal{F}_i[x_j := 1][x_k := 0][x_l := 1]$ with
184 respect to ordering π'
185 $a^1 := a^1_1 ... a^1_{i+2} c^1$
186 $b^1 := b^1_1 ... b^1_{i+2} 11...1$
187 $a^2 := a^2_1 ... a^2_{i+2} c^2$
188 $b^2 := b^2_1 ... b^2_{i+2} 11...1$
189 $a^3 := a^3_1 ... a^3_{i+2} c^3$
190 **for all** $k \leq n$ unmapped by π **do**
191 $\pi(k) = \pi'(k - i - 2)$
192 **done**
193 **output** ordering π and intervals $[a^1, b^1], [a^2, b^2]$ and $[a^3, 2^n - 1]$
194 **halt**
195 **else**
196 **output NO**
197 **endif**

Now we are ready to prove the correctness of Algorithm 2. Part of this proof is actually the proof of correctness of algorithm recognizing positive 2-interval functions and is taken from [4] with some adjustments so we can continue with the recognition of 3-interval functions.

Theorem 5.5.11. *Algorithm 2 correctly recognizes positive* 3*-interval functions on n variables in prime and irredundant DNF and can be implemented to run in* $O(n.l)$ *time, where l is the length of the given DNF.*

Proof. The first part (steps 6 - 23) of Algorithm 2 is in fact the algorithm recognizing positive 1-interval functions. If the algorithm outputs 0 or an ordering

and one interval, then the correctness of this output follows from the correctness of that algorithm (Theorem 5.2.3). If this part terminates with a nonempty DNF, then due to Lemma 5.5.1 the input function is not a 1-interval function and the parts of π and a^1, which has been constructed so far, are correct in the sense, that if there is a 2-interval representation of the input function f, then it can be constructed by extending this π and a^1. This is justified by Corollary 5.4.3, because in the first part of the algorithm only variables forming linear terms or occurring in every term are taken as the most significant ones. If the resulting DNF $\mathcal{F}_i$ is not empty, then (according to Corollary 5.5.3) it represents a 2-interval function if and only if it has pair of variables x_i, x_j defining the form $\widetilde{\mathcal{S}}_2^3$ and embedded functions are 1-interval with respect to the same ordering of the remaining variables. Due to Lemma 5.4.19 we can choose any pair, which defines the form $\widetilde{\mathcal{S}}_2^3$.

If the previous condition is not fulfilled, Lemma 5.5.2 tells us that the given function is not 2-interval. At this point, Corollary 5.5.7 guarantees that $\mathcal{F}_i$ is 3-interval if and only if it has exactly one of the forms $\widetilde{\mathcal{S}}_2^3$, $\widetilde{\mathcal{S}}_2^4$, $\widetilde{\mathcal{S}}_3^4$, 5.6 or 5.7 and the embedded functions satisfy the respective condition as described in Lemma 5.5.4. In any of these cases we will again look for set of defining (leading, resp.) variables and Lemmas 5.4.19, 5.5.8 and 5.5.9 ensure us that we can choose any such set.

Afterwards, we need to check if the embedded functions are 1-interval (or in one case one of them 2-interval). In all cases, where we have to check several functions if they are 1-interval with respect to the same ordering of their variables, we can use several instances of the algorithm recognizing positive 1-interval functions running in parallel.

How to implement such parallel executions is described in [2]. Specifically, in the implementation of the algorithm recognizing general 1-interval functions. That parallel implementation is designed for two parallel runs, however, it is not hard to see that it can be used for three runs as well. It maintains the set I, which is intersection of sets I_1 and I_2 of variables possible to choose in current step of respective runs. Such set can be in the same way maintained for three runs. Each time when we add a new suitable variable into one of the sets I_1, I_2, I_3, we can in constant time (according to the implementation in [2]) check if the variable is present in both of the other sets, and if so, add it also to the intersection I.

However, as we already analyzed above, the situation gets more complicated, if we want to run in parallel one algorithm for recognition of 1-interval functions and other one for 2-interval functions. We will proceed as follows.

At first, we will run the algorithms for both these functions separately to determine if they are both 1-interval only with respect to a different ordering, or one of them is 2-interval such that it is not 1-interval. This is done in $O(l)$ time.

In the case that one of them is 2-interval (let it be $\mathcal{F}_2$ and consequently $\mathcal{F}_1$ is 1-interval), we will run the parallel implementation as above until we get empty intersection I. Since $\mathcal{F}_1$ is 1-interval, its set I_1 is not empty and if the shared

ordering exists, one of the variables from I_1 has to be the most significant among unprocessed variables. Otherwise $\mathcal{F}_1$ would have at least 2-intervals with respect to any other ordering, because it satisfies conditions of Lemma 5.5.10.

The 2-interval function $\mathcal{F}_2$ has one of the forms $\widetilde{\mathcal{S}}_1^2, \widetilde{\mathcal{S}}_2^2$ or $\widetilde{\mathcal{S}}_2^3$. In the first two cases it has its own non-empty set I_2, which is disjoint with I_1 and therefore satisfies the conditions of Lemma 5.5.10. Specifically, fixing any variable from I_1 will break function $\mathcal{F}_2$ into two non-constant functions.

If $\mathcal{F}_2$ has form $\widetilde{\mathcal{S}}_2^3$, then fixing any variable will result in two non-constant functions as well. This is ensured by pair of Lemmas 5.4.8 and 5.4.13 or by Lemma 5.4.12 depending on which variable is fixed and if the fixation makes one of the embedded functions constant.

Therefore, for each $x_i \in I_1$ we will try to finish the execution of three parallel algorithms for recognition 1-interval functions $\mathcal{F}_1[x_j := a], \mathcal{F}_2[x_j := 0]$ and $\mathcal{F}_2[x_j := 1]$, where a is appropriate value according to the algorithm. This will be finished in $O(n.l)$ time. If the shared ordering exists, one of these executions will find it, because in this way, we try all possibilities.

The situation is a little different, when we have two 1-interval functions. In this case when intersection I becomes empty, both I_1 and I_2 are non-empty, but disjoint. In this case we will simply try all variables from both sets and finish three parallel executions as described above. In each case, according to the choice of variable, one of the functions will be in the role of 1-interval function and the second one will break into two 1-interval functions according to the Lemma 5.5.10. This will again take $O(n.l)$ time.

The last thing is to show, how to find all five 3-interval forms in $O(n.l)$ time. In the following text T will be the number of terms in DNF $\mathcal{F}_i$.

To identify the pair of defining variables of form $\widetilde{\mathcal{S}}_2^3$ (on line 25), we need to go through the DNF once, identify quadratic terms and count occurrences of every variable. Then we go through all the pairs of variables forming quadratic terms and choose the first pair, for which the sum of the occurrences of both variables is equal to $T + 1$ (because of the occurrences in the quadratic term formed by such pair).

The situation is similar with identification of the triplet of variables for form $\widetilde{\mathcal{S}}_3^4$ (line 118). We go through the DNF once, identify terms of length three and count occurrences of every variable. We also check if all terms are of length at least three. Then we go through all the triplets of variables forming terms of length three and choose the first one, for which the sum of the occurrences of variables is equal to $2T + 1$ and each variable has at least three occurrences.

The detection of form $\widetilde{\mathcal{S}}_2^4$ (line 94) is different. If we have a candidate triplet of variables, such that there are tree quadratic terms with them, then we can in constant time check if they are the correct ones. We simply check if the sum of their occurrences is $T + 3$ and each one has at least three occurrences. If we choose any quadratic term (during the counting walk through the DNF), at least one of its variables has to be one of the variables of the correct triplet. Therefore,

for both these variables we simply try all $\binom{n}{2}$ combinations with the rest of the variables until we find the correct set. This takes in total $O(n^2 + l)$, which is also $O(n.l)$.

In order to detect the leading and co-leading variables of form 5.6 on the line 143, we go through the DNF once, identify quadratic terms and count the occurrences of variables in DNF and occurrences of variables in quadratic terms and for each variable create the linked list of terms in which it occurs. Then we go through all variables and check those, which satisfy following conditions. They have two occurrences in DNF and two terms in its linked list are quadratic. Hence they define two candidates for co-leading variables. For each such variable then we go through the DNF once more and check if each term contains at least one of the co-leading variables. This is obviously done in $O(n.l)$ time.

The leading and the co-leading variables of form 5.7 (line 169) can be detected as follows. At first, we perform the counting walk through the DNF and pick an arbitrary quadratic term. Then we go through the list of variables and look for the variable, which is not in this term and has $T - 1$ occurrences. If we find such variable, we have our triplet of leading variables. If not, then one of the variables of our quadratic term has to be the leading variable (otherwise we conclude that $\mathcal{F}_i$ does not have the form 5.7). We check the number of occurrences of these variables, go once through the DNF and look for the terms without one of these variables. If for one of them holds, that it has $T - 1$ occurrences and we found quadratic term without it, we have found the triplet of leading variables. This is done in $O(n + l)$ time, which is also $O(l)$.

When we summarize the time of all analyzed procedures we will get overall time of execution $O(n.l)$. ∎

Chapter 6

General k-Interval Functions

6.1 $\texttt{PARITY}_n$ function

As we already know, that $\texttt{PARITY}_n$ and $\texttt{PARITY}_n \oplus 1$ are commutative functions. We will now look closer into their structure and we will calculate the number of their intervals.

Theorem 6.1.1. *Function* $\texttt{PARITY}_n$ *has round*$(\frac{2^n}{3})$ *truepoint intervals and* $\lceil \frac{2^n}{3} \rceil$ *falsepoint intervals for any ordering of its variables.*

Proof. Since $\texttt{PARITY}_n \oplus 1$ is negation of $\texttt{PARITY}_n$, it is easy to observe, that $\texttt{PARITY}_n$ has equal truepoint intervals as $\texttt{PARITY}_n \oplus 1$ has falsepoint intervals and vice versa. Therefore it is sufficient to calculate only the number of truepoint intervals for both these functions.

We will denote number of truepoint intervals for $\texttt{PARITY}_n$ as P_n and for $\texttt{PARITY}_n \oplus 1$ as $\overline{P}_n$.

Let us now analyze, what happens if we fix one of the variables in $\texttt{PARITY}_n$ function. Without loss of generality we choose x_n. We know, that $\texttt{PARITY}_n = \texttt{PARITY}_{n-1} \oplus x_n$. Hence $\texttt{PARITY}_n[x_n := 0] \Leftrightarrow \texttt{PARITY}_{n-1} \oplus 0$ which is equivalent to $\texttt{PARITY}_{n-1}$ and $\texttt{PARITY}_n[x_n := 1] \Leftrightarrow \texttt{PARITY}_{n-1} \oplus 1$.

Therefore, if we want to construct the projection interval of $\texttt{PARITY}_n$, we need to concatenate the projection intervals of $\texttt{PARITY}_{n-1}$ and $\texttt{PARITY}_{n-1} \oplus 1$ in this particular order.

Now we need to analyze what are the values of leftmost and rightmost points of projection interval of studied functions. $\texttt{PARITY}_n(\boldsymbol{0}) = 0$ and $\texttt{PARITY}_n \oplus 1(\boldsymbol{0}) = 1$ for any n. n-dimensional vector $\boldsymbol{1}$ has even amount of ones if and only if n is even. Therefore from the definition of $\texttt{PARITY}_n$ function we have that $\texttt{PARITY}_{n-1}(\boldsymbol{1}) = 1$ and $\texttt{PARITY}_{n-1} \oplus 1(\boldsymbol{1}) = 0$ if and only if n is even.

Therefore for each $k \in \{1, ...\}$ following equalities hold:

$$n = 2k \Rightarrow P_n = P_{n-1} + \overline{P_{n-1}} - 1 \tag{6.1a}$$

$$n = 2k + 1 \Rightarrow P_n = P_{n-1} + \overline{P_{n-1}} \tag{6.1b}$$

In other words, if we concatenate projections for even n, then the projection of $\texttt{PARITY}_{n-1}$ ends with truepoint interval, which will be merged with starting truepoint interval of $\texttt{PARITY}_{n-1} \oplus 1$ and therefore we will loose one interval.

Moreover if n is odd, then $\texttt{PARITY}_n$ has equal number of truepoint intervals as falsepoint intervals as a direct corollary of the fact that the projection interval starts in 0 and ends in 1. Similarly for even n the projection starts in 0 and also ends in 0, therefore it has one more falsepoint interval than it has truepoint intervals. Since $\overline{P_n}$ is also the number of falsepoint intervals of $\texttt{PARITY}_n$, we can write:

$$n = 2k \Rightarrow \overline{P_n} = P_n + 1 \tag{6.2a}$$

$$n = 2k + 1 \Rightarrow \overline{P_n} = P_n \tag{6.2b}$$

If we put equations 6.1 and 6.2 together, we will get following:

$$P_n = 2P_{n-1} + (-1)^{n-1}$$

This equation can be be iteratively decomposed until we get the summation starting in $n = 0$:

$$\begin{aligned}
P_n &= 2P_{n-1} + (-1)^{n-1} \\
P_n &= 4P_{n-2} + 2^1(-1)^{n-2} + 2^0(-1)^{n-1} \\
P_n &= 8P_{n-3} + 2^2(-1)^{n-3} + 2^1(-1)^{n-2} + 2^0(-1)^{n-1} \\
&\ldots \\
P_n &= \sum_{k=0}^{n-1} 2^k(-1)^{n-1-k} = (-1)^{n-1}\sum_{k=0}^{n-1}(-2)^k \\
P_n &= (-1)^{n+1}\frac{(-2)^n - 1}{-3} = \frac{2^n}{3} - \frac{(-1)^n}{3} = round\left(\frac{2^n}{3}\right)
\end{aligned}$$

We already have the number of truepoint intervals for $\texttt{PARITY}_n$. We can further observe, that for even n we round $\frac{2^n}{3}$ to the next smaller integer and for odd n we round it to the next greater integer. And since for even n we have one more falsepoint interval as we have truepoint intervals, we can conclude that the number of falsepoint intervals for $\texttt{PARITY}_n$ is $\lceil \frac{2^n}{3} \rceil$. ∎

6.2 Number of intervals for general k-interval functions

In paper [3] the following hypothesis is presented as Theorem 1.5.

Hypothesis 3. *Any boolean function on n variables has at most $2^{n-1} = \frac{2^n}{2}$ intervals. Moreover, this bound is tight.*

The proof suggests as an example a function, which alternate zeros and ones on every neighboring vectors in given ordering and claims that $\texttt{PARITY}_n$ is such function. However, we already know that $\texttt{PARITY}_n$ does not look like this and also has less intervals as proved in Theorem 6.1.1, namely $\frac{2^n}{3}$ instead of $\frac{2^n}{2}$. This alternating of values actually happens for any $i \in \{1, ..., n\}$ on n-ary function $f = x_i$ for any ordering of variables where x_i is the least significant one. And we can easily observe, that this function is 1-interval function for any ordering of variables where x_i is the most significant one. Therefore, there are Boolean functions on n variables that have for certain ordering of their variables 2^{n-1} intervals, however, this bound is not tight.

We believe that a more exact proposition about the number of intervals for general k-interval functions can be formulated. However, we do not have the proof for it, therefore we will present it as another hypothesis.

Hypothesis 4. *Let f be a k-interval Boolean function on n variables such that $f \in \mathcal{C}_{k-int} \setminus \mathcal{C}_{(k-1)-int}$. Then $k \leq \lceil \frac{2^n}{3} \rceil$.*

Chapter 7

Conclusion

In the thesis we studied mainly positive k-interval functions. We discovered some of their forms and analyzed their properties. We formulated various propositions and we used them in the new recognition algorithm for positive 3-interval functions. However, many of these propositions are generalized for positive k-interval functions and can be utilized in further studies of such functions and possible recognition algorithms for $k \geq 4$.

It the would be useful to prove Hypothesis 2 in order to make any progress in this area for $k \geq 4$. However, it will definitely not be enough. Since the forms $\widetilde{\mathcal{S}}_k^m$ are not the only forms of positive Boolean functions, it is required to discover and analyze also other forms. It is possible to begin with those forms $\widetilde{\mathcal{S}}_k^m[0]$ and $\widetilde{\mathcal{S}}_k^m[1]$, which according to Observation 5.4.15 do not degenerate into forms $\widetilde{\mathcal{S}}_k^m$. Unfortunately, it still might not be enough. As we could see, positive 3-interval functions can have also forms 5.6 and 5.7 and it is appropriate to assume that positive k-interval functions for $k \geq 4$ will also have some other forms. Therefore, it might be necessary to somehow generalize such forms and analyze their properties.

When we consider time complexity of possible algorithm for recognition of positive k-interval functions, where $k \geq 4$, it is sufficient to consider the case, when such function has the form $\widetilde{\mathcal{S}}_2^3$ and one of embedded functions is 3-interval. It is obvious, that k will get into the exponent at least at this point. It is possible, that the exponent might get only to n.

Finally, the thesis studied the structure and number of intervals of functions `MAJORITY` and `PARITY`. We also formulated the Hypothesis 1 and 4 that these functions have the highest tight bound amount of intervals among positive and general boolean functions respectively. We assume that the proofs would require to show, that if the given function has for some ordering of its variables more intervals, it should be possible to find (using the branching tree operators) the ordering with respect to which the given function has less or the same number of intervals as the respective one of these functions.

Bibliography

[1] Crama, Y., and Hammer, P. L.: *Boolean Functions: Theory, Algorithms, and Applications* (2008).

[2] Čepek, O., Kronus, D., and Kučera, P.: *Recognition of Interval Boolean Functions*, Annals of Mathematics and Artificial Intelligence, Volume 52, Number 1 (2008), 1-24.

[3] Kronus, D.: *Relations of Treshold and k-Interval Boolean Functions*, Tech. rep., Charles University in Prague, Dep. of Theoretical Computer Science (2007).

[4] Kronus, D., and Čepek, O.: *Recognition of Positive 2-Interval Boolean Functions*, Tech. rep., Charles University in Prague, Dep. of Theoretical Computer Science (2007).

[5] Chandra, A.-K., and Iyengar, V.-S.: *Constraint solving for test case generation a technique of high level design verification*, Proceedings of the IEEE International Conference on Computer Design (1992).

[6] Garey, M., and Johnson, D.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco (1979).

[7] Schieber, B., Geist, D., and Ayal Z.: *Computing the minimum dnf representation of boolean functions defined by intervals*, Discrete Applied Mathematics (2005), 149, 154 - 173.

[8] Barwise, J.: Handbook of Mathematical Logic (Studies in Logic and the Foundations of Mathematics), North Holland (1982).

[9] Quine, W.: *The problem of simplifying the truth functions*, Amer. Math. Monthly 59 (1952), 521 - 531.

[10] Genesereth, M., and Nilsson, N.: *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, Los Altos, CA, (1987).

[11] Lewin, D., Fournier, L., Roytman, E., and Shurek, G.: *Constraint satisfaction for test program generation*, Proceedings of the 14th IEEE International Phoenix Conference on Computers and Communications (1995), 45 – 48.

[12] DeMillo, R.-A., and Offutt, A.-J.: *Constraint-based automatic test data generation*, IEEE Trans. on Software Engng. 17 (1991), 900 – 910.

[13] Tinder, R. F.: *Engineering Digital Design, Second Edition*, Elsevier Science, USA (2000).